Ph.D. DISSERTATION

# Learning to Rank Texts for Question Answering System Using Deep Neural Network

## 질의응답 시스템을 위한 텍스트 랭킹 심층 신경망

BY

SEUNGHYUN YOON

AUGUST 2020

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

# Learning to Rank Texts for Question Answering System Using Deep Neural Network

## 질의응답 시스템을 위한 텍스트 랭킹 심층 신경망

BY

SEUNGHYUN YOON

AUGUST 2020

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

# Learning to Rank Texts for Question Answering System Using Deep Neural Network

질의응답 시스템을 위한 텍스트 랭킹 심층 신경망

지도교수 정 교 민
이 논문을 공학박사 학위논문으로 제출함

2020년 8월

서울대학교 대학원

전기 · 정보 공학부

윤 승 현

윤승현의 공학박사 학위 논문을 인준함

2020년 8월

| | |
|---|---|
| 위 원 장: | 심 규 석 |
| 부위원장: | 정 교 민 |
| 위 원: | 윤 성 로 |
| 위 원: | 김 건 희 |
| 위 원: | 차 미 영 |

# Abstract

The question answering (QA) system has attracted huge interests due to its applicability in real-world applications. This dissertation proposes novel ranking algorithms for the QA system based on deep neural networks. We first tackle the long-text QA that requires the model to understand the excessively large sequence of text inputs. To solve this problem, we propose a hierarchical recurrent dual encoder that encodes texts from word-level to paragraph-level. We further propose a latent topic clustering method that utilizes semantic information in the target corpus, and thus it increases the performance of the QA system. Secondly, we investigate the short-text QA, where the information in text pairs are limited. To overcome the insufficiency, we combine a pretrained language model and an enhanced latent clustering method to the QA model. This novel architecture enables the model to utilizes additional information, resulting in achieving state-of-the-art performance for the standard answer-selection tasks (i.e., WikiQA, TREC-QA). Finally, we investigate detecting supporting sentences for complex QA system. As opposed to the previous studies, the model needs to understand the relationship between sentences to answer the question. Inspired by the hierarchical nature of the text, we propose a graph neural network-based model that iteratively propagates necessary information between text nodes and achieve the best performance among existing methods.

**keywords**: ranking texts, question answering system, deep neural network

**student number**: 2017-36633

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Recently neural network architectures have shown great success in many machine learning fields such as image classification, speech recognition, machine translation, response generation, question answering, and other task-oriented areas. Among these, the automatic question answering (QA) system has long been considered a primary objective of artificial intelligence. The advancement of the QA system has attracted huge interests from the academic and industry community for several reasons. First, it provides the service at any time, which increases the users' satisfaction. Second, it can be easily scaled-out by launching the service on the cloud infrastructure. Furthermore, it provides consistent information to users that are unfeasible through the conventional human-agent system.

Question answering is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language [10]. The QA system uses structured and unstructured data to provide answers to user questions. Intelligent voice agents such as "Bixby" of Samsung Electronics and "Siri" of Apple use pre-defined structural data. When these services receive a query from the user, they try to match it with pre-defined actions and execute that action. Another type of research analyzes user questions and retrieves relevant information from

**Passage:**

Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. When asked where all the money had gone, Tesla responded by saying that he was affected by the ***Panic of 1901***, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Q: On what did Tesla blame for the loss of the initial money?

A: Panic of 1901

Figure 1.1: An example of machine reading QA.

indexed data to provide results to the user. The "AnswerBus" system implements a QA system based on sentence-level Web information retrieval by using several search engines and achieves 70.5% accuracy for factoid type of question [11]. More recent QA systems are using both structured and unstructured data after analyzing user queries. The "IBM Watson" system extracts structured information from unstructured data and builds a knowledge base [12]. It also indexes unstructured data using search clusters [13].

Building the whole QA system requires the integration of different technologies such as information retrieval, knowledge construction, ranking, answer-verification, answer generation, and other algorithms of computer science discipline. Among these sub-QA systems, our research objectives lie in investigating neural network-based models for ranking question-answer pairs in the QA system. As the goal of a ranking system is to find the best answers to the question among the candidates, it can be formulated as a computing matching-score between the text (i.e., question and an answer candidate). This type of QA problem can be categorized into three groups in regard to the "answer-span" it considers. First, we can consider a task to find an exact answer from a passage to a given question. Figure 1.1 shows an example of a machine

Passage:
① Journey to the West is one of the four classics of Chinese literature. ② Written by the Ming Dynasty novelist Wu Cheng'en during the 16th century, this beloved adventure tale combines action, humor, and spiritual lessons.

③ The novel takes place in the seventh century. ④ It tells the story of one of Buddha Sakyamuni's disciples who was banished from the heavenly paradise for the crime of slighting the Buddha Law. ⑤ He was sent to the human world and forced to spend ten lifetimes practicing religious self-cultivation in order to atone for his sins.

*⑥ In his tenth lifetime, now during the Tang Dynasty, he reincarnates as a monk named Xuan Zang (also known as Tang Monk and Tripitaka).* ⑦ The emperor wishes this monk can travel west and bring holy Mahayana Buddhist scriptures back to China. ⑧ After being inspired by a vision from the Bodhisattva Guanyin, the monk accepts the mission and sets off on the sacred quest.

Q: Who is the Tang?

A: ⑥

Figure 1.2: An example of sentence-level answer candidates.

reading QA dataset. Each word in the passage can be an answer candidate; thus, the QA system tries to find the exact answer-span, i.e., start- and end- index of the word from the passage that can answer the question. After introducing several benchmark dataset for this task [14, 15, 16, 17, 18, 19, 20], researchers proposed various neural network-based model on this tasks [21, 22, 23, 24, 25]. In another group, we can consider a task to find a sentence-level answer for the given question. Figure 1.2 shows an example of sentence-level answer candidates. The sentences are extracted from the passage and considered as answer candidates. A model is trained to rank the sentence that contains the exact answer in it, higher than other sentences. This type of task, matching similarity score between two sentences (question and answer candidate), has been widely investigated in the research community [1, 26, 27, 28, 29]. There exist a fundamental framework to tackle these tasks [1, 3] and further development upon previous works [2, 4, 30, 31, 32, 33]. Recently, researchers integrate the pretrained language model into the tasks [6, 24, 34], and apply the transfer learning technique to increase the model performance [35, 36].

Passage:

① Journey to the West is one of the four classics of Chinese literature. Written by the Ming Dynasty novelist Wu Cheng'en during the 16th century, this beloved adventure tale combines action, humor, and spiritual lessons.

② The novel takes place in the seventh century. It tells the story of one of Buddha Sakyamuni's disciples who was banished from the heavenly paradise for the crime of slighting the Buddha Law. He was sent to the human world and forced to spend ten lifetimes practicing religious self-cultivation in order to atone for his sins.

*③ In his tenth lifetime, now during the Tang Dynasty, he reincarnates as a monk named Xuan Zang (also known as Tang Monk and Tripitaka). The emperor wishes this monk can travel west and bring holy Mahayana Buddhist scriptures back to China. After being inspired by a vision from the Bodhisattva Guanyin, the monk accepts the mission and sets off on the sacred quest.*

Q: Who is the Tang?

A: ③

Figure 1.3: An example of lengthy answer candidates.

Apart from these approaches, we can consider lengthy answer candidates, as shown in Figure 1.3 [9, 37]. In this case, paragraph- or passage-level of the answer is ranked and provided to the user [38, 39, 40, 41, 42, 43]. As a model deals with lengthy information in the answer candidates, it is likely to provide a more accurate answer to the user. At the same time, the user can check the evidence by reading the provided answer.

Among these types of QA problems, we investigate the lengthy answer candidate case first. To tackle this task, we proposed a neural network-based QA-pair ranking model through the following research:

- Seunghyun Yoon, Joongbo Shin, and Kyomin Jung, "Learning to Rank Questionanswer Pairs Using Hierarchical Recurrent Encoder with Latent Topic Clustering," in *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, New Orleans, USA, June 2018.

This study proposes a hierarchical recurrent dual encoder model that encodes passage

to a vector representation from a word-level to a chunk-level to capture the entire meaning effectively. By adapting the hierarchical structure, the HRDE shows very small performance degradations in lengthy text comprehension while other state-of-the-art recurrent neural network models suffer from it. Besides, this study proposes a latent topic clustering module that extracts semantic information from target samples. This clustering module is useful for any text related tasks by allowing each data sample to find its nearest topic cluster, thus helping the neural network model to analyze the entire data. We evaluate our models on the Ubuntu Dialogue Corpus and consumer electronic domain question answering dataset, which is related to Samsung products. The proposed model shows state-of-the-art results for ranking question-answer pairs. As our research show successful performance in computing matching score between two texts (i.e., a question and lengthy answer candidate), we extend the research to rank a news article in regards with the incongruity between the headline and body content of news articles [44]. Our experiments and qualitative evaluations demonstrate that the proposed methods outperform existing approaches and efficiently detect news stories with misleading headlines in the real world.

After success in building a model for ranking long-passage-type of QA pairs, we move on to one of the fundamental QA ranking tasks, an answer-selection task, which aims to rank short-sentence type of answer candidates, (as described in Figure 1.2). This task is more complicated than the previously studied long-passage ranking task since the information of the given text gets decreased as the length of the text is lessened. There exists a standard framework, *Compare Aggregate framework* [1], that computes a matching score between the question-answer pair via token-wise matching. We enhance this framework and propose a new model through the following research:

- Seunghyun Yoon, Franck Dernoncourt, Doo Soon Kim, Trung Bui, and Kyomin Jung, "A Compare-Aggregate Model with Latent Clustering for Answer Selection," in *Proceedings of the 28th ACM International on Conference on Information and Knowledge Management (CIKM)*, Beijing, China, November 2019.

In this study, we explore the effect of additional information by adopting a pretrained language model to compute the vector representation of the input text and by applying transfer learning from a large-scale corpus. Then, we enhance the compare-aggregate model by integrating a latent clustering method to compute additional information within the target corpus and by changing the objective function from listwise to pointwise. To evaluate the proposed approaches, we conduct a series of experiments with the WikiQA and TREC-QA datasets. The empirical results demonstrate the superiority of our proposed approach, which achieves state-of-the-art performance for both datasets.

Next, we investigate a model that detects supporting sentences for question answering. As opposed to the previous studies, the model needs to understand the relationship between sentences and to extract necessary information from each sentence to answer the question. It is a very challenging task since previous answer-selection models do not consider the relational information across the passage. To fill the gap, we propose a novel graph neural network that propagates information over sentences to understand information that cannot be inferred when considering sentences in isolation.

- Seunghyun Yoon, Franck Dernoncourt, Doo Soon Kim, Trung Bui, and Kyomin Jung, "Propagate-Selector: Detecting Supporting Sentences for Question Answering via Graph Neural Networks,"in *Proceedings of The 12th International Conference on Language Resources and Evaluation (LREC)*, Marseille, France, May 2020.

In this study, we design a graph structure in which each node represents an individual sentence, and some pairs of nodes are selectively connected based on the text structure. Then, we develop an iterative attentive aggregation and a skip-combine method in which a node interacts with its neighborhood nodes to accumulate the necessary information. To evaluate the performance of the proposed approaches, we conduct experiments with the standard HotpotQA dataset. The empirical results demonstrate the superiority of our proposed approach, which obtains the best performances, compared

to the widely used answer-selection models that do not consider the inter-sentential relationship.

The remainder of this dissertation is organized as follows. Chapter 2 provides a background on textual data representation in natural language processing. In chapter 3, we explain the proposed question-answer pair ranking methods for lengthy text. Chapter 4 explains a model for the answer-selection task of the short sentence. Further investigation of supporting sentence detection for complex question answering is discussed in Chapter 5. Finally, the dissertation is concluded in Chapter 6.

# Chapter 2

# Background

In the dissertation, we present our investigations, i.e., algorithms and models that learn to rank text pairs for question answering system. In particular, our research focuses on a deep neural network-based model that recently shows significant performance improvements in many machine learning areas. In building a model that efficiently learns the pattern in the text data for the natural language processing (NLP) tasks, it is essential to understand how the model deals with discrete information in the text, i.e., the sequence of words. In this chapter, we introduce two widely used techniques in the NLP applications; they act as fundamental sub-blocks in developing a model.

## 2.1 Textual Data Representation

For NLP applications, including question answering system, words or phrases from the vocabulary are mapped to vectors of real numbers; we call it word-embedding. As word-embedding is the first step in the NLP model that converts discrete text-form data to feature-level, it primarily affects the overall performance of the model. There have been many studies on neural network-based language models for word-level representations.

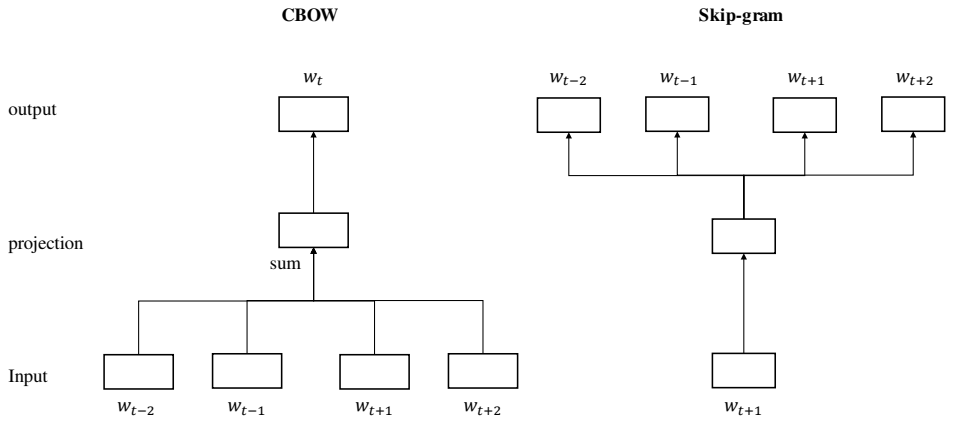



Figure 2.1: The Continuous Bag-of-Words (CBOW) architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. [5]

### Distributed word representation

Distributed word representations were proposed and gained huge interests as they were considered to be fundamental building blocks for the natural language processing tasks [45, 46, 47].

The distributed word representation can be assigned by considering the placement of each word in a sentence. The basic intuition is "A word is characterized by the company it keeps" [48]. Mikolov et al. (2013) propose two novel model architectures for computing continuous vector representations of words from very large data sets [47]. Figure 2.1 shows the two methods, continuous bag-of-words (CBOW) and skip-gram, that compute the distributed representation of a word based on the surrounding context in a sentence. This representation, called *word2vec*, shows high performance in measuring syntactic and semantic word similarities; thus, it is used as an initialized value in the word-embedding layer in many NLP models. These approaches enable unsupervised training on a huge corpus. On the other hand, the representation of a word is mapped to the same value regardless of the specific context. For example, this model

assigns the same value to "*present*" in the following two sentences, even though they have a different meaning. "We have a collection for Henry's retirement *present*," "The play is set in the *present*."

In another line of study, Pennington et al. (2014) proposed GloVe that is an unsupervised learning algorithm for obtaining vector representations for words [49]. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. This model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrences matrix, rather than on the entire sparse matrix or individual context windows in a large corpus.

### Contextualized representations

Recently, researchers explored contextualized representations of text where each word will have different representations depends on the context [6, 24, 34]. This idea is motivated by neural text classification tasks. Given a sentence $X = \{x_1, ..., x_t\}$, where $x_t$ is a *t*-th word in a sentence, one can build a neural network-based model that classify the sentence to some category *Y*. A deep neural network model, (e.g., Long short term memory (LSTM) [50] or Gated recurrent units (GRU) [51], is stacked upon the word embedding to compute a sequence of hidden states as follows:

$$h_t = \text{LSTM}(h_{t-1}, e(x_t)), \tag{2.1}$$

where $h_t$ is *t*-th hidden state of the LSTM and $e$ denotes word embedding function. Then feed the averaged hidden states to the classifier to compute the categorical probabilities as follows:

$$\begin{aligned} h_{\text{avg}} &= \text{average}([h_1, ..., h_t]), \\ P(Y = y \mid \mathbf{X}) &= \frac{\exp(W_y^\intercal h_{\text{avg}})}{\sum_{y' \in Y} \exp(W_{y'}^\intercal h_{\text{avg}})}, \end{aligned} \tag{2.2}$$

where $W$ is learned model weight.

In the model above, we regard the $h_t$ as a contextualized representation of $x_t$. To compute a contextualized representation of each word in a sentence for general purpose, researchers combined the aforementioned idea with an unsupervised training objective. Peters et al. (2018) proposed a pre-trained language model, Embeddings from Language Model (ELMo). This research adopts the language model as its objective function to train the model. It employs an LSTM to encode the context and construct the next-word distribution as follows:

$$P(x_t \,|\, \mathbf{X}_{<t}) = \frac{\exp(e_\theta(x_t)^\intercal h_{t-1}}{\sum_{x' \in V} \exp(e_\theta(x')^\intercal h_{t-1})}, \tag{2.3}$$

where $h_{t-1}$ is the last hidden state of the LSTM that encodes $\mathbf{X}_{1:t-1}$, $e_\theta$ is word embedding layer, $V$ is set of vocabulary (i.e., word). In particular, the research employs forward- and backward-LSTM and concatenate their outputs to model the sequential pattern in a text in both direction. Then the model can be trained with maximum likelihood learning objective as follows:

$$\max_{\theta} \; \log P_\theta(\mathbf{X}) = \sum_{t=1}^{T} \log P_\theta(x_t | \mathbf{X}_{<t}). \tag{2.4}$$

Note that this approach is fully unsupervised since the label is "next-word" in a sentence. Figure 2.2 shows the overall architecture of the ELMo. The model first map each word to vector representation in the word-embedding layer; then forward and backward long short term memory (LSTM) [50] are employed to encode the sequence of words in a sentence. Finally, three vector representations from the aforementioned components (word-embedding, forward- and backward-LSTM) are merged to construct the final contextual representations of each word.

In another line of research, Radford et al. (2018) employed the transformer architecture [8] to build a deep language model trained with large unlabeled text corpora [34]. More recently, Devlin et al. (2019) proposed the bidirectional encoder representations from Transformers (BERT) [24] with the masked language modeling(MLM) objective, which is to predict the original ids of explicitly masked words from the input.

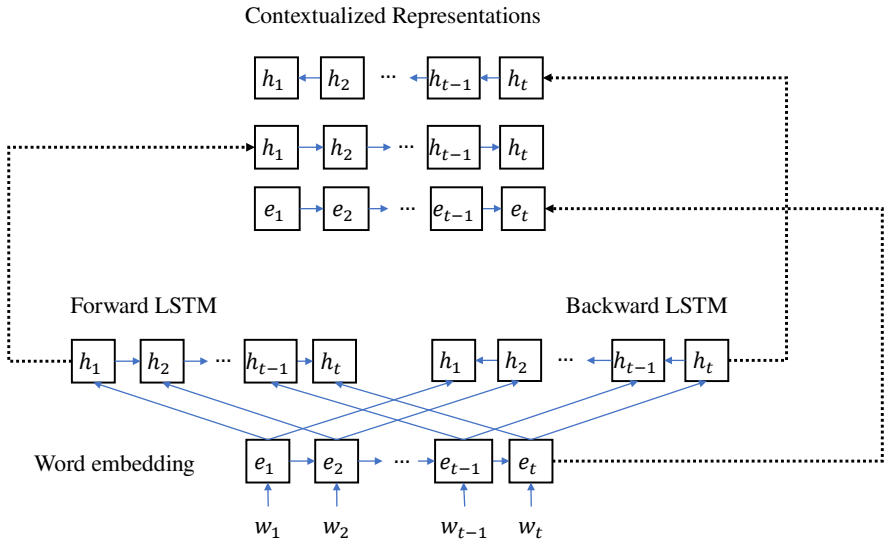



Figure 2.2: Model architecture of the deep contextualized word representations [6]

Due to the MLM objective, each contextual word representation should be computed by a two-step process of masking the word in the input and feeding it into the BERT.

These aforementioned contextualized representations and its variations have brought significant improvements in learning natural language representation, and they have achieved the state-of-the-art performances on various downstream tasks such as GLUE benchmark [52] and question answering [14]. However, the use of contextualized word representation to unsupervised tasks, (e.g., N-best list reranking for automatic speech recognition and neural machine translation), still remains challenging.

## 2.2 Encoding Sequential Information in Text

In this section, we introduce several techniques for dealing with sequential information in a text. When encoding word sequence in a text, recurrent neural network (RNN)-, convolutional neural network (CNN)-based encoding approaches have been widely used. More recently, a self-attention network (SAN) is proposed.

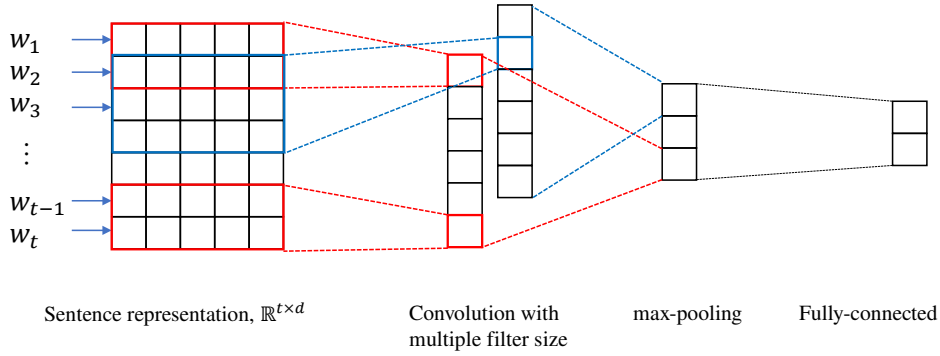



Figure 2.3: Model architecture of CNN-based word representations [7]

**RNN-based encoding skim**

A subset of sequential data, e.g., words in a sentence, is fed into RNN which updates the internal hidden state $h_t$ to model the time series patterns as follows:

$$\begin{aligned} h_t &= f_\theta(h_{t-1}, e_t), \\ e_t &= g_\theta(w_t), \end{aligned} \tag{2.5}$$

where $f_\theta$ is the RNN function, e.g., LSTM or GRU, with weight parameter $\theta$, $h_t$ is internal hidden state at *t*-th word input, $w_t$ is *t*-th word in a target sentence, $e_t$ is vector representation mapped to $w_t$ using word-embedding representations, $g_\theta$ (see the section 2.1). In computing the sentence representation, last hidden state, $h_{\text{last}}$, is considered to contain all the information in the sentence. In a different way, each hidden state can be weighted-, averaged, or max-pooled to form a sentence representation.

**CNN-based encoding skim**

Unlike RNN, CNN architecture does not have any *memory* process that is capable of accumulating the information from the sequence data. To fill this gap, Kim (2014) [7] proposed CNN-based architecture trained on top of pre-trained word vectors for sentence-level classification tasks [7]. In this model, each CNN channel aggregates information from the vectorial word representation. In particular, various channels with a different

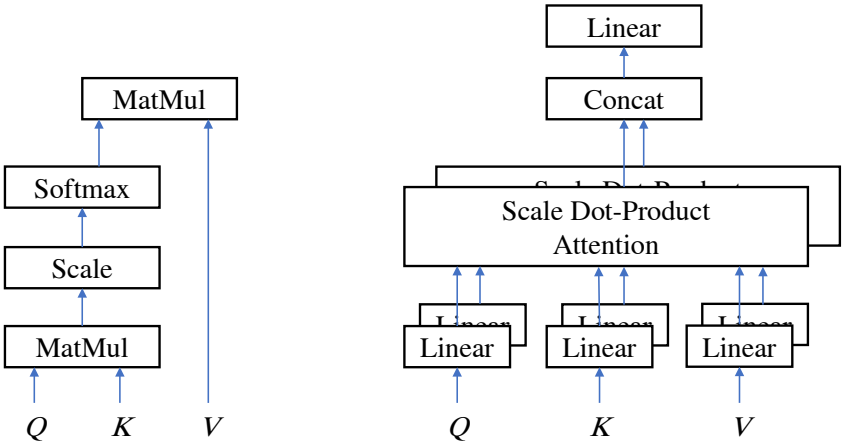



Figure 2.4: Model architecture of "selt-attention network (left)" and "multi-head attention (right)". [8]

size enable the model to deal with *n*-gram-like information in a sentence. As shown in the Figure 2.3, this model takes the word sequence of any sentence as input to the convolutional layer; then, it obtains a vector representation $\boldsymbol{v} = \{v_i | i = 1, \cdots, k\}$ for each part of the sentence through the max-over-time pooling after computing convolution with $k$ filters as follows:

$$v_i = g(f_i(W)), \tag{2.6}$$

where $g$ is max-over-time pooling function, $f_i$ is the CNN function with *i*-th convolutional filter, and $W \in \mathbb{R}^{t \times d}$ is a matrix of the word sequence.

**Self-attention network based encoding skim**

Recently, self-attention network based [8] architectures achieve significant performance improvement in various NLP tasks. This architecture employs "scale-dot-product attention" and "multi-head attention" mechanisms to encode textual information. Figure 2.4 shows these two architectures. In scale-dot-product attention algorithm, atten-

tion is computed as follows:

$$\text{Scaled-dot-product-Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.7}$$

where *Q*, *K*, *V* represents query, key, and value, respectively. $d_k$ is the dimension of the *K*.

The multi-head attention mechanism is defined as follows:

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1, ..., \text{head}_n)W^O, \\ &\text{head}_i = \text{Scaled-dot-product-Attention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \tag{2.8}$$

where *W* indicates the trainable weight matrix for each component (i.e., Q, K, and V). With this architecture, each word in a text will be updated with the weighted summing information, among other words in a sentence. Naturally, we can assume that the first word will contain all the representative information of the sentence after passing through the multiple stacked attention layers. Devilin et al. (2019) [24] put a special token at the first position of every sentence; then, consider it as the representative token of the sentence.

In the literature, researchers adopt one of these techniques (or hybrid some of these approaches) to encode a sentence. Note that all of these techniques provide both sentence-level and word-level representations. According to the specific applications, these representations will be selectively applied.

# Chapter 3

# Question-Answer Pair Ranking for Long Text

Recently neural network architectures have shown great success in many machine learning fields such as image classification, speech recognition, machine translation, chat-bot, question answering, and other task-oriented areas. Among these, the automatic question answering (QA) task has long been considered a primary objective of artificial intelligence.

In the commercial sphere, the QA task is usually tackled by using pre-organized knowledge bases and/or by using information retrieval (IR) based methods, which are applied in popular intelligent voice agents such as *Siri*, *Alexa*, and *Google Assistant* (from Apple, Amazon, and Google, respectively). Another type of advanced QA systems is IBM's Watson who builds knowledge bases from unstructured data. These raw data are also indexed in search clusters to support user queries [12, 13].

In academic literature, researchers have intensely studied sentence pair ranking task which is core technique in QA system. The ranking task selects the best answer among candidates retrieved from knowledge bases or IR based modules. Many neural network architectures with end-to-end learning methods are proposed to address this task [53, 1, 54]. These works focus on matching sentence-level text pair [26, 55, 56]. Therefore, they have limitations in understanding longer text such as multi-turn dialogue and explanatory document, resulting in performance degradation on ranking as

the length of the text become longer.

With the advent of the huge multi-turn dialogue corpus [9], researchers have proposed neural network models to rank longer text pair [57, 58]. These techniques are essential for capturing context information in multi-turn conversation or understanding multiple sentences in explanatory text.

In this paper, we focus on investigating a novel neural network architecture with additional data clustering module to improve the performance in ranking answer candidates which are longer than a single sentence. This work can be used not only for the QA ranking task, but also to evaluate the relevance of next utterance with given dialogue generated from the dialogue model. The key contributions of our work are as follows:

First, we introduce a Hierarchical Recurrent Dual Encoder (HRDE) model to effectively calculate the affinity among question-answer pairs to determine the ranking. By encoding texts from an word-level to a chunk-level with hierarchical architecture, the HRDE prevents performance degradations in understanding longer texts while other state-of-the-art neural network models suffer.

Second, we propose a Latent Topic Clustering (LTC) module to extract latent information from the target dataset, and apply these additional information in end-to-end training. This module allows each data sample to find its nearest topic cluster, thus helping the neural network model analyze the entire data. The LTC module can be combined to any neural network as a source of additional information. This is a novel approach using latent topic cluster information for the QA task, especially by applying the combined model of HRDE and LTC to the QA pair ranking task.

Extensive experiments are conducted to investigate efficacy and properties of the proposed model. Our proposed model outperforms previous state-of-the-art methods in the Ubuntu Dialogue Corpus, which is one of the largest text pair scoring datasets. We also evaluate the model on real world QA data crawled from crowd-QA web pages and from Samsung's official web pages. Our model also shows the best results for the

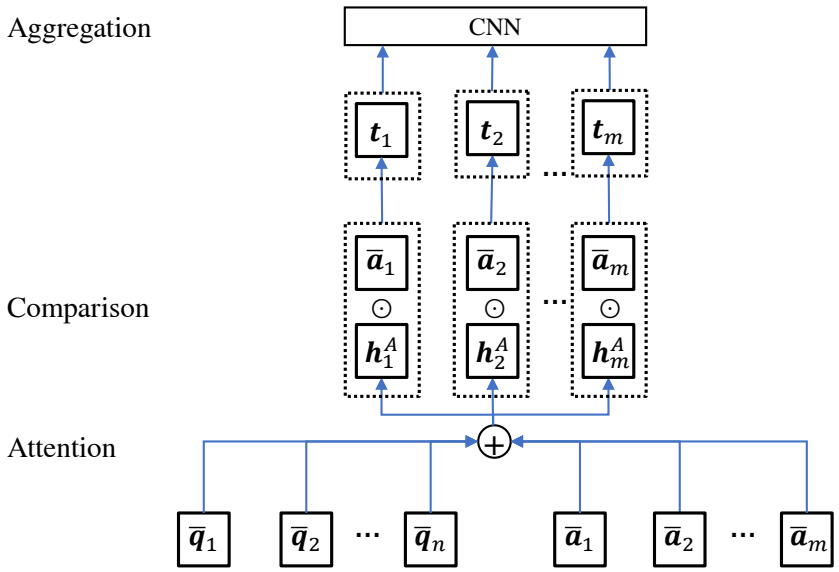



Figure 3.1: Diagram of the compare aggregate framework [1].

QA data when compared to previous neural network based models.

## 3.1 Related Work

Researchers have released question and answer datasets for research purposes and have proposed various models to solve these datasets. [26, 55, 59] introduced small dataset to rank sentences that have higher probabilities of answering questions such as WikiQA and insuranceQA. To alleviate the difficulty in aggregating datasets, that are large and have no license restrictions, some researchers introduced new datasets for sentence similarity rankings [58, 9]. As of now, the Ubuntu Dialogue dataset is one of the largest corpus openly available for text ranking.

To tackle the Ubuntu dataset, [9] adopted the "term frequency-inverse document frequency" approach to capture important words among context and next utterances [60]. [61, 62] proposed deep neural network architecture for embedding sentences and measuring similarities to select answer sentence for a given question. [57] used

convolution neural network (CNN) architecture to embed the sentence while a final output vector was compared to the target text to calculate the matching score. They also tried using long short-term memory (LSTM) [50], bi-directional LSTM and ensemble method with all of those neural network architectures and achieved the best results on the Ubuntu Dialogues Corpus dataset. Another type of neural architecture is the RNN-CNN model, which encodes each token with a recurrent neural network (RNN) and then feeds them to the CNN [58]. Researchers also introduced an attention based model to improve the performance [1, 54, 59]. In particular, Compare Aggregate Framework [1], as shown in Figure 3.1, inspired many other works.

Recently, the hierarchical recurrent encoder-decoder model was proposed to embed contextual information in user query prediction and dialogue generation tasks [63, 64]. A lower-level RNN embedded sentence level information from sequence inputs of the words in each sentence while an upper-level RNN embedded sentence turns level information from the sequence input of the lower-level RNN output vector. This shows improvement in the dialogue generation model where the context for the utterance is important. As another type of neural network architecture, memory network was proposed by [65]. Several researchers adopted this architecture for the reading comprehension (RC) style QA tasks, because it can extract contextual information from each sentence and use it in finding the answer [66, 67]. However, none of this research is applied to the QA pair ranking task directly.

## 3.2 Method

### 3.2.1 Baseline Approach

#### Recurrent Dual Encoder (RDE)

Recurrent neural network (RNN) is a variant of neural network which is designed to learn sequential or time-varying patterns [68]. A subset of sequential data is fed into the recurrent neural network (RNN) which leads to the formation of the network's

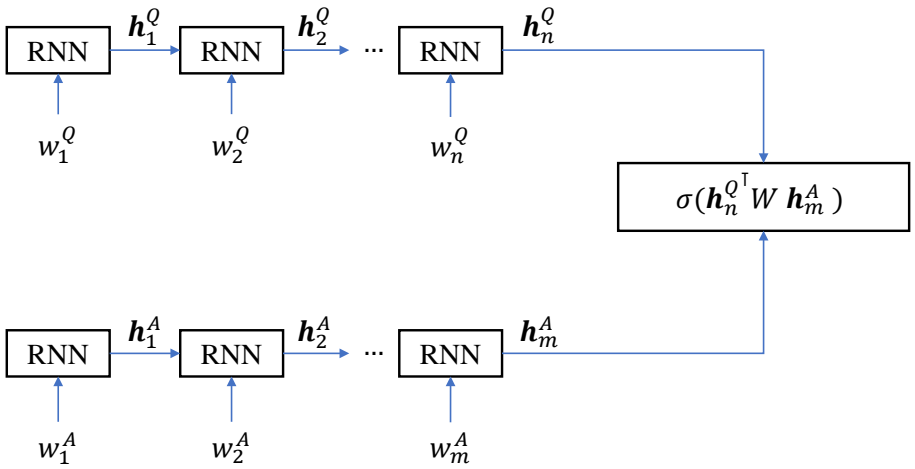



Figure 3.2: Diagram of the recurrent dual encoder model [9]. Each RNN encodes words sequence in question, $w^Q$, and answer, $w^A$. The matching score is retrieved by using dot-product calculation.

internal hidden state $h_t$ to model the time series patterns. This internal hidden state is updated at each time step with the input data $w_t$ and the hidden state of the previous time step $h_{t-1}$ as follows:

$$h_t = f_\theta(h_{t-1}, w_t), \tag{3.1}$$

where $f_\theta$ is the RNN function with weight parameter $\theta$, $h_t$ is hidden state at *t*-th word input, $w_t$ is *t*-th word in a target question $\boldsymbol{w}^Q = \{w^Q_{1:t_q}\}$ or an answer text $\boldsymbol{w}^A = \{w^A_{1:t_a}\}$ .

The previous RDE model uses two RNNs for encoding question text and answer text to calculate affinity among texts [9], as shown in Figure 3.2. After encoding each part of the data, the affinity among the text pairs is calculated by using the final hidden state value of each question and answer RNNs. The matching probability between

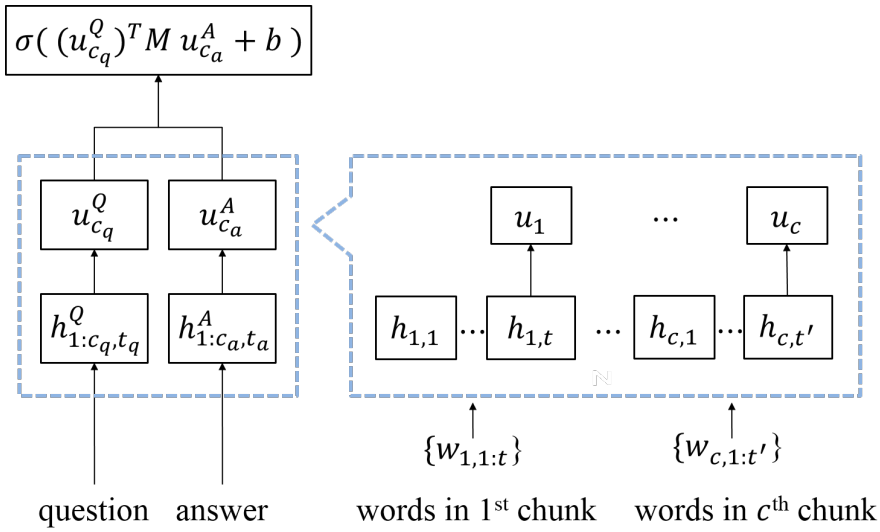



Figure 3.3: Diagram of the HRDE model. The word-lever RNN encodes words sequences of each chunk. The the final hidden status of the word-level RNN is fed into chunk-level RNN.

question text $\boldsymbol{w}^Q$ and answer text $\boldsymbol{w}^A$ with the training objective are as follows:

$$
\begin{aligned}
p(\text{label}) &= \sigma((h^Q_{t_q})^T M\ h^A_{t_a} + b), \\
\mathcal{L} &= -\log \prod_{n=1}^{N} p(\text{label}_n | h^Q_{n,t_q}, h^A_{n,t_a}),
\end{aligned}
\tag{3.2}
$$

where $h^Q_{t_q}$ and $h^A_{t_a}$ are last hidden state of each question and answer RNN with the dimensionality $h_t \in \mathbb{R}^d$. The $M \in \mathbb{R}^{d \times d}$ and bias $b$ are learned model parameters. The $N$ is total number of samples used in training and $\sigma$ is the sigmoid function. The matching probability also can be calculated by using another feed-forward neural network model as in [69], however we uses the dot-product method since it provides similar performance with less computation cost.

### 3.2.2 Proposed Approaches (HRDE+LTC)

#### Hierarchical Recurrent Dual Encoder (HRDE)

From now we explain our proposed model [38]. The previous RDE model tries to encode the text in question or in answer with RNN architecture. It would be less effective as the length of the word sequences in the text increases because RNN's natural characteristic of forgetting information from long ranging data. To address this RNN's forgetting phenomenon, [70] proposed an attention mechanism, however, we found that it still showed a limitation when we consider very large sequential length data such as 162 steps average in the Ubuntu Dialogue Corpus dataset (see Table 3.3). To overcome this limitation, we designed the HRDE architecture. The HRDE model divides long sequential text data into small chunk such as sentences, and encodes the whole text from word-level to chunk-level by using two hierarchical level of RNN architecture.

Figure 3.3 shows a diagram of the HRDE model. The word-level RNN part is responsible for encoding the words sequence $\boldsymbol{w}_c = \{w_{c,1:t}\}$ in each chunk. The chunk can be sentences in paragraph, paragraphs in essay, turns in dialogue or any kinds of smaller meaningful sub-set from the text. Then the final hidden states of each chunk will be fed into chunk-level RNN with its original sequence order kept. Therefore the chunk-level RNN can deal with pre-encoded chunk data with less sequential steps. The hidden states of the hierarchical RNNs are as follows:

$$\begin{aligned} h_{c,t} &= f_\theta(h_{c,t-1}, w_{c,t}), \\ u_c &= g_\theta(u_{c-1}, h_c), \end{aligned} \tag{3.3}$$

where $f_\theta$ and $g_\theta$ are the RNN function in hierarchical architecture with weight parameters $\theta$, $h_{c,t}$ is word-level RNN's hidden status at *t*-th word in *c*-th chunk. The $w_{c,t}$ is *t*-th word in *c*-th chunk of target question or answer text. The $u_c$ is chunk-level RNN's hidden state at *c*-th chunk sequence, and $h_c$ is word-level RNN's last hidden state of each chunk $h_c \in \{h_{1:c,t}\}$. For the choice of RNN function, we use the Gated Recurrent

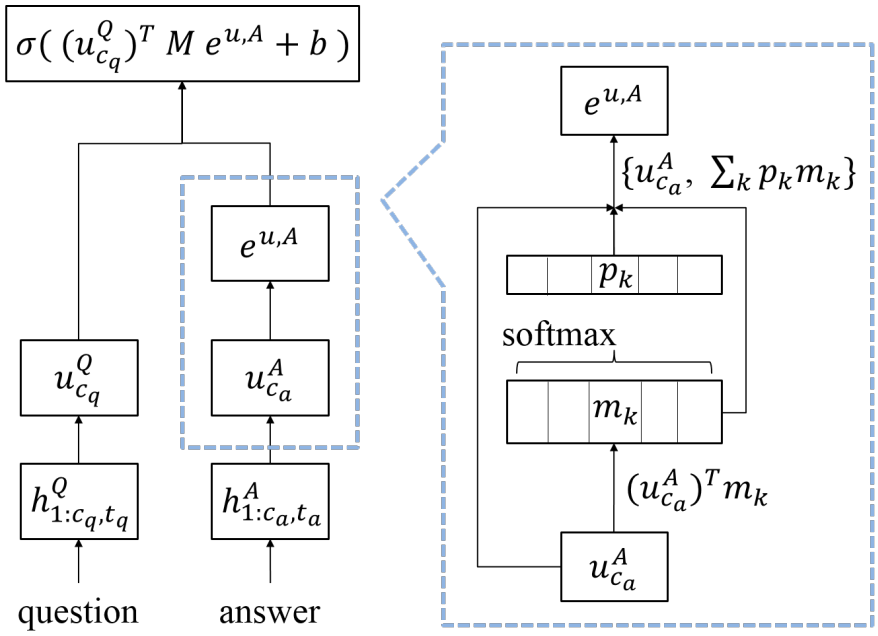



Figure 3.4: Diagram of the HRDE-LTC. Input vector is compared to each latent topic memory $m_k$ to calculate cluster-info contained vector. This vector will be concatenated to original input vector.

Unit (GRU) because it shows compatible performance to the LSTM while requiring less weight parameters [51].

We use the same training objective as the RDE model, and the final matching probability between question and answer text is calculated using chunk-level RNN as follows:

$$p(\text{label}) = \sigma((u_{c_q}^{Q})^T M\ u_{c_a}^{A} + b), \tag{3.4}$$

where $u_{c_q}^{Q}$ and $u_{c_a}^{A}$ are chunk-level RNN's last hidden state of each question and answer text with the dimensionality $u_c \in \mathbb{R}^{d^u}$, which involves the $M \in \mathbb{R}^{d^u \times d^u}$.

**Latent Topic Clustering (LTC)**

To learn how to rank QA pairs, a neural network should be trained to find the proper feature that represents the information within the data and fits the model parameter that can approximate the true-hypothesis. For this type of problem, we propose the LTC module for grouping the target data to help the neural network find the true-hypothesis with more information from the topic cluster in end-to-end training.

The blue-dotted box on the right-side of Figure 3.4 shows LTC structure diagram. To assign topic information, we build internal latent topic memory $m \in \mathbb{R}^{d^m \times K}$, which is only model parameter to be learned, where $d^m$ is vector dimension of each latent topic and $K$ is number of latent topic cluster. For a given input sequence $\boldsymbol{x} = \{x_{1:t}\}$with these K vectors, we construct LTC process as follows:

$$
\begin{aligned}
p_k &= \text{softmax}((\boldsymbol{x})^T m_k), \\
\boldsymbol{x}_K &= \sum_{k=1}^{K} p_k m_k, \\
\boldsymbol{e} &= \text{concat}\{\boldsymbol{x}, \boldsymbol{x}_K\}.
\end{aligned}
\tag{3.5}
$$

First, the similarity between the $\boldsymbol{x}$ and each latent topic vector is calculated by dot-product. Then the resulting $K$ values are normalized by the softmax function $\text{softmax}(z_k) = e^{z_k}/\sum_i e^{z_i}$ to produce a similarity probability $p_k$. After calculating the latent topic probability $p_k$, $x_K$ is retrieved from summing over $m_k$ weighted by the $p_k$. Then we concatenate this result with the original encoding vector to generate the final encoding vector $\boldsymbol{e}$ with the LTC information added.

Note that the input sequence of the LTC could be any type of neural network based encoding function $\boldsymbol{x} = f_\theta^{\text{enc}}(\boldsymbol{w})$ such as RNN, CNN and multilayer perceptron model (MLP). In addition, if the dimension size of $\boldsymbol{x}$ is different from that of memory vector, additional output projection layer should be placed after $\boldsymbol{x}$ before applying dot-product to the memory.

**Combined Model of (H)RDE and LTC**

As the LTC module extracts additional topic cluster information from the input data, we can combine this module with any neural network in their end-to-end training flow. In our experiments, we combine the LTC module with the RDE and HRDE models.

**RDE with LTC:** The RDE model encodes question and answer texts to $h^Q_{t_q}$ and $h^A_{t_a}$, respectively. Hence, the LTC module could take these vectors as the input to generate latent topic cluster information added vector $\boldsymbol{e}$. With this vector, we calculate the affinity among question and answer texts as well as additional cluster information. The following equation shows our RDE-LTC process:

$$p(\text{label}) = \sigma((h^Q_{t_q})^T M\ e^A + b). \tag{3.6}$$

In this case, we applied the LTC module only for the answer side, assuming that the answer text is longer than the question. Thus, it needs to be clustered. To train the network, we use the same training objective, to minimize cross-entropy loss, as in equation (3.2).

**HRDE with LTC:** The LTC can be combined with the HRDE model, in the same way it is applied to the RDE-LTC model by modifying equation (3.6 as follows:

$$p(\text{label}) = \sigma((u^Q_{c_q})^T M\ e^{u,A} + b), \tag{3.7}$$

where $u^Q_{c_q}$ is the final network hidden state vector of the chunk-level RNN for a question input sequence. The $e^{u,A}$ is the LTC information added vector from equation (3.5), where the LTC module takes the input $\boldsymbol{x} = \boldsymbol{u}^A$ from the HRDE model equation (3.3). The HRDE-LTC model also use the same training objective, minimizing cross-entropy loss, as in equation (3.2). Figure 3.4 shows a diagram of the combined model with the HRDE and the LTC.

Table 3.1: Example of the Ubuntu-v2 dataset

| **Question** | **Answer** | *flag* |
|---|---|---|
| “if i want to instal ubuntu 12.04 on a ( to be creat ) raid1+lvm partit , can the setup creat those for me or do i have to do it separ ?” “if you use the altern cd instal you can . ca n’t with the livecd *unless* you do it manual” “ok. what about the “ server ” iso then ? can it do that too ?” | “yes , it ’s the main user of those facil” | 1 |
| | “http : //www.eclipse.org/downloads/moreinfo/c.php” | 0 |
| | “i dident even see your origin problem” | 0 |
| | “so whi do you want to know ... ? : ) ( cron log to /var/log/messag by default , but it also mail you output from your job as long as you ve instal an mta )” | 0 |
| | “and you re right , often those common walnut size brain ’ scenario be the one that keep you bang your head to the wall until you realiz that : “ wait a minut ! what if ... be you use synapt by chanc ? ” lol” | 0 |

## 3.3 Experimental Setup and Dataset

### 3.3.1 Dataset

#### The Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus has been developed by expanding and preprocessing the Ubuntu Chat Logs[1], which refer to a collection of logs from the Ubuntu-related chat room for solving problem in using the Ubuntu system by [9].

The data was preprocessed by replacing named entities with generic tags such as

[1]These logs are available from http://irclogs.ubuntu.com

names, locations, organizations, URLs, and system paths. The data was further processed extracting (context, response, flag) triples from dialogues. Among the utterances in the dialogues, they consider each utterance, starting from the third one, as a potential {response} while the previous utterance is considered as a {context}. The data was processed extracting ({context}, {response}, flag) tuples from the dialogues. These tuples are used as ground-truth data, and applied negative sampling to the response to generate *false*-label data for training.

Table 3.2: Example of the Consumer Product QA dataset

| **Question** |
| --- |
| “how do i manage sent messages in message folders ?” |

| **Answer** | *flag* |
| --- | --- |
| ”from the messaging menu , select sent . your list of sent messages appear in the display . 2 highlight a message in the list . you have the following options : to remove the highlighted message from the sent folder , press erase . to review the highlighted message , press open . for more options , press options . the following options appear in the display : – forward : send the highlighted message to another recipient . – resend : send another copy of the highlighted message to the original recipient . – add to contacts : save the recipient ’ s number to your contacts . – message info : show message delivery status , addressee , size , and so on . – manage sent : choose to erase this message , to lock the message from being erased , or to unlock this message to be erased . – sort by recipient / time : list the sent messages in order of their recipient ’ s number or the time the messages were sent . drafts 3 compose your message using the keypad , press messages in the draft folder are those that have been options , and then select save as draft . “ message composed but never sent . you can return to the draft folder saved ” appears in the display , and your message is at any time to view , edit , or send a draft message . saved to the drafts folder .” | 1 |
| “although the phone does not have a configurable pop 3 or imap 4 application , there is a mobile web service that features mobile email . this feature provides the opportunity to receive ” new mail ” alerts with shortcut access to your inbox to read , delete , and respond to your e-mail as if you were on your pc . to access the e-mail from your browser , follow the steps below : from the standby screen , press the right softkey for browser select email and networking select from the following email options : using navigation keys , login by entering username and password and select sign in .” | 0 |

Table 3.3: Properties of the Ubuntu and Consumer Product QA dataset. The message and response are {context}, {response} in Ubuntu and {question}, {answer} in the Consumer Product QA dataset

| Dataset | # Samples | | | Message (Avg.) | | | Response (Avg.) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Train | Val. | Test | # tokens | # groups | # tokens /group | # tokens | # groups | # tokens /group |
| **Ubuntu-v1** | 1M | 35,609 | 35,517 | 162.47 $\pm$132.47 | 8.43 $\pm$6.32 | 20.14 $\pm$18.41 | 14.44 $\pm$13.93 | 1 | - |
| **Ubuntu-v2** | 1M | 19,560 | 18,920 | 85.92 $\pm$74.71 | 4.95 $\pm$2.98 | 20.73 $\pm$20.19 | 17.01 $\pm$16.41 | 1 | - |
| **Consumer product QA** | 163,616 | 10,000 | 10,000 | 12.84 $\pm$6.42 | 1 | - | 173.48 $\pm$192.12 | 6.09 $\pm$5.58 | 29.28 $\pm$31.91 |

We called this original Ubuntu dataset as Ubuntu-v1 dataset. After releasing the Ubuntu-v1 dataset[2], researchers published v2 version of this dataset[3]. Main updates are separating train/valid/test dataset by time so that mimics real life implementation, where we are training a model on past data to predict future data, changing sampling procedure to increase average turns in the {context}. We consider this Ubuntu dataset is one of the best dataset in terms of its quality, quantity and availability for evaluating the performance of the text ranking model. In our experiments with the Ubuntu-v1 dataset, we use the same preprocessing, i.e. tokenization and named entities replacement, and vocabulary size for fair comparison. For the Ubuntu-v2 dataset, we use standard preprocessing of the data using the python-based natural language toolkit NLTK [71]. We perform tokenization only to see the model performance clearly.

To encode the text with the HRDE and HRDE-LTC model, a text needs to be divided into several chunk sequences with predefined criteria. For the Ubuntu-v1 dataset case, we divide the {context} part by splitting with end-of-sentence delimiter "_eos_", and we do not split the {response} part since it is normally short and does not contain "_eos_" information. For the Ubuntu-v2 dataset case, we split the {context} part in the same way as we do in the Ubuntu-v1 dataset while only using end-of-turn delimiter "_eot_". Table 3.3 shows properties of the Ubuntu dataset and Table 3.1 shows examples of the dataset.

### 3.3.2 Consumer Product Question Answering Corpus

To test the robustness of the proposed model, we introduce an additional question and answer pair dataset, named Consumer Product QA Corpus (CP-QA), that is related to an actual user's interaction with the consumer electronic product domain. We crawled data from various sources like the Samsung Electronics' official web site[4] and crowd

[2]http://dataset.cs.mcgill.ca/ubuntu-corpus-1.0

[3]https://github.com/rkadlec/ubuntu-ranking-dataset-creator

[4]http://www.samsung.com/us

QA web sites[5,6] in a similar way that [72] did in building QA system for consumer products. On the official web page, we can retrieve data consisting of user questions and matched answers like frequently asked questions and troubleshooting. From the crowd QA sites, there are many answers from various users for each question. Among these answers, we choose answers from company certificated users to keep the reliability of the answers high. If there are no such answers, we skip that question answer pair. Table 3.2 shows an example of question-answer pair crawled from the web page. In addition, we crawl hierarchical product category information related to QA pairs. In particular, *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance* as top-level categories, and specific categories like *galaxy s7*, *tablet*, *led tv*, and *others* are used. We collected these meta-information for further use. The total size of the CP-QA data is over 100,000 pairs and we split the data into approximately 80,000/10,000/10,000 samples to create train/valid/test sets, respectively. To create the train set, we use a QA pair sample as a ground-truth and perform negative sampling for answers among training sets to create *false*-label datasets. In this way, we generated ({question}, {answer}, flag) triples (see Table 3.3). We do the same procedure to create valid and test sets by only differentiating more negative sampling within each dataset to generate 9 *false*-label samples with one ground-truth sample. We apply the same method in such a way that the Ubuntu dataset is generated from the Ubuntu Dialogue Corpus to maintain the consistency. To encoding the text with the HRDE and HRDE-LTC model, we group the answer text by applying NLTK sentence tokenizer, and we do not group the question text because it it normally short. Table 3.3 contains properties of the CP-QA dataset. (the consumer product question answering corpus). Table 3.2 shows examples of the CP-QA dataset.

[5]http://answers.yahoo.com

[6]http://answers.us.samsung.com

### 3.3.3 Implementation Details

#### Ubuntu dataset case

To implement the RDE model, we use two single layer Gated Recurrent Unit (GRU) [51] with 300 hidden units . Each GRU is used to encode {context} and {response}, respectively. The weight for the two GRU are shared. The hidden units weight matrix of the GRU are initialized using orthogonal weights [73], while input embedding weight matrix is initialized using a pre-trained embedding vector, the Glove [49], with 300 dimension. The vocabulary size is 144,953 and 183,045 for the Ubuntu-v1/v2 case, respectively. We use the Adam optimizer [74], with gradients clipped with norm value 1. The maximum time step for calculating gradient of the RNN is determined according to the input data statistics in Table 3.3.

For the HRDE model, we use two single layer GRU with 300 hidden units for word-level RNN part, and another two single layer GRU with 300 hidden units for chunk-level RNN part. The weight of the GRU is shared within the same hierarchical part, word-level and chunk-level. The other settings are the same with the RDE model case. As for the combined model with the (H)RDE and the LTC, we choose the latent topic memory dimensions as 256 in both ubuntu-v1 and ubuntu-v2. The number of the cluster in LTC module is decided to 3 for both the RDE-LTC and the HRDE-LTC cases. In HRDE-LTC case, we applied LTC module to the {context} part because we think it is longer having enough information to be clustered with. All of these hyper-parameters are selected from additional parameter searching experiments.

The dropout [75] is applied for the purpose of regularization with the ratio of: 0.2 for the RNN in the RDE and the RDE-LTC, 0.3 for the word-level RNN part in the HRDE and the HRDE-LTC, 0.8 for the latent topic memory in the RDE-LTC and the HRDE-LTC.

We need to mention that our implementation of the RDE module has the same architecture as the LSTM model [57] in ubuntu-v1/v2 experiments case. It is also the

Table 3.4: Model performance results for the Ubuntu-v1 dataset

| Model | Ubuntu-v1 | | | |
|---|---|---|---|---|
| | 1 in 2 R@1 | 1 in 10 R@1 | 1 in 10 R@2 | 1 in 10 R@5 |
| **TF-IDF** [9] | 0.659 | 0.410 | 0.545 | 0.708 |
| **CNN** [57] | 0.848 | 0.549 | 0.684 | 0.896 |
| **LSTM** [57] | 0.901 | 0.638 | 0.784 | 0.949 |
| **CompAgg** [1] | 0.884 | 0.631 | 0.753 | 0.927 |
| **BiMPM** [54] | 0.897 | 0.665 | 0.786 | 0.938 |
| **RDE** | 0.898 $\pm$0.002 | 0.643 $\pm$0.009 | 0.784 $\pm$0.007 | 0.945 $\pm$0.002 |
| **RDE-LTC** | 0.903 $\pm$0.001 | 0.656 $\pm$0.003 | 0.794 $\pm$0.003 | 0.948 $\pm$0.001 |
| **HRDE** | 0.915 $\pm$0.001 | 0.681 $\pm$0.001 | 0.820 $\pm$0.001 | 0.959 $\pm$0.001 |
| **HRDE-LTC** | **0.916** $\pm$0.001 | **0.684** $\pm$0.001 | **0.822** $\pm$0.001 | **0.960** $\pm$0.001 |

same architecture with the RNN model [58] in ubuntu-v2 experiment case. We implement the same model ourselves, because we need a baseline model to compare with other proposed models such as the RDE-LTC, HRDE and HRDE-LTC.

### Customer Product QA dataset case

To test the CP-QA dataset, we use the same implementation of the model (RDE, RDE-LTC, HRDE and HRDE-LTC) used in testing the Ubuntu dataset. Only the differences are, we use 100 hidden units for the RDE and the RDE-LTC, 300 hidden units for the HRDE and 200 hidden units for the HRDE-LTC, and the vocabulary size of 28,848. As for the combined model with the (H)RDE and LTC, the dimensions of the latent topic memory is 64 and the number of latent cluster is 4. We chose best performing hyper-parameter of each model by additional extensive hyper-parameter search experiments. All of the code developed for the empirical results are available via web repository[7].

[7] http://github.com/david-yoon/QA_HRDE_LTC

Table 3.5: Model performance results for the Ubuntu-v2 dataset

| Model | Ubuntu-v2 | | | |
|---|---|---|---|---|
| | 1 in 2 R@1 | 1 in 10 R@1 | 1 in 10 R@2 | 1 in 10 R@5 |
| **LSTM** [9] | 0.869 | 0.552 | 0.721 | 0.924 |
| **RNN** [58] | 0.907 ±0.002 | 0.664 ±0.004 | 0.799 ±0.004 | 0.951 ±0.001 |
| **CNN** [58] | 0.863 ±0.003 | 0.587 ±0.004 | 0.721 ±0.005 | 0.907 ±0.003 |
| **RNN-CNN** [58] | 0.911 ±0.001 | **0.672** ±0.002 | 0.809 ±0.002 | 0.956 ±0.001 |
| **RNN-CNN-Attn** [59] | 0.903 ±0.002 | 0.653 ±0.005 | 0.788 ±0.005 | 0.945 ±0.002 |
| **CompAgg** [1] | 0.895 | 0.641 | 0.776 | 0.937 |
| **BiMPM** [54] | 0.877 | 0.611 | 0.747 | 0.921 |
| **RDE** | 0.894 ±0.002 | 0.610 ±0.008 | 0.776 ±0.006 | 0.947 ±0.002 |
| **RDE-LTC** | 0.899 ±0.002 | 0.625 ±0.004 | 0.788 ±0.004 | 0.951 ±0.001 |
| **HRDE** | 0.914 ±0.001 | 0.649 ±0.001 | 0.813 ±0.001 | 0.964 ±0.001 |
| **HRDE-LTC** | **0.915** ±0.002 | 0.652 ±0.003 | **0.815** ±0.001 | **0.966** ±0.001 |

## 3.4 Empirical Results

We regards all the tasks as selecting the best answer among text candidates for the given question. Following the previous work [9], we report model performance as recall at $k$ (R@k) relevant texts among given 2 or 10 candidates (e.g., 1 in 2 R@1). Though this metric is useful for ranking task, R@1 metric is also meaningful for classifying the best relevant text.

Each model we implement is trained multiple times (10 and 15 times for Ubuntu and the CP-QA datasets in our experiments, respectively) with random weight initialization, which largely influences performance of neural network model. Hence we report model performance as mean and standard derivation values (Mean$\pm$Std).

Table 3.6: Model performance results for the Consumer Product QA dataset

| Model | Consumer Product QA | | | |
|---|---|---|---|---|
| | 1 in 2 R@1 | 1 in 10 R@1 | 1 in 10 R@2 | 1 in 10 R@5 |
| **TF-IDF** | 0.939 | 0.834 | 0.897 | 0.953 |
| **RDE** | 0.978 $\pm 0.002$ | 0.869 $\pm 0.009$ | 0.966 $\pm 0.003$ | 0.997 $\pm 0.001$ |
| **RDE-LTC** | 0.981 $\pm 0.002$ | 0.880 $\pm 0.009$ | 0.970 $\pm 0.003$ | 0.997 $\pm 0.001$ |
| **HRDE** | 0.981 $\pm 0.002$ | 0.885 $\pm 0.011$ | 0.971 $\pm 0.004$ | 0.997 $\pm 0.001$ |
| **HRDE-LTC** | **0.983** $\pm 0.002$ | **0.890** $\pm 0.010$ | **0.972** $\pm 0.003$ | **0.998** $\pm 0.001$ |

### 3.4.1 Comparison with other methods

As Table 3.4 shows, our proposed HRDE and HRDE-LTC models achieve the best performance for the Ubuntu-v1 dataset. We also find that the RDE-LTC model shows improvements from the baseline model, RDE.

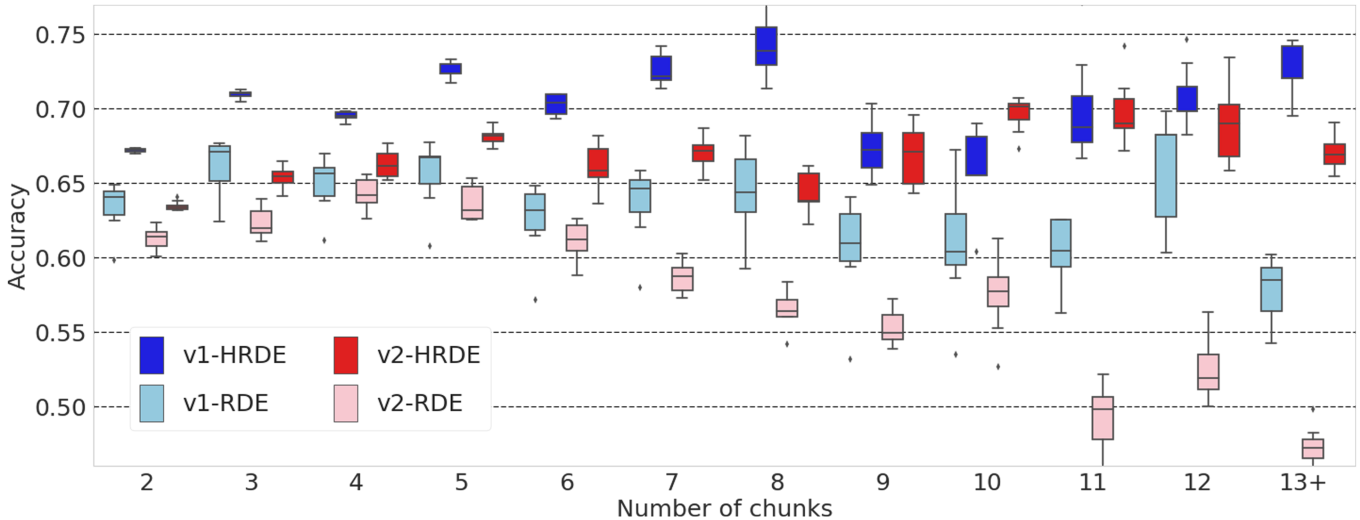



Figure 3.5: The HRDE and RDE model performance comparisons for the number-of-chunk in the Ubuntu dataset. Each boxplot shows average accuracy with standard deviation. The HRDE models, in darker blue and red colors, show consistent performances as the number-of-chunks increased. Meanwhile, the RDE models in lighter colors show performance degradation as the number-of-chunks increased. Furthermore, 13+ indicates all data over 13-chunks.

For the ubuntu-v2 dataset case, Table 3.5 reveals that the HRDE-LTC model is best for three cases (1 in 2 R@1, 1 in 10 R@2 and 1 in 10 R@5). Comparing the same model with our implementation (RDE) and [58]'s implementation (RNN), there is a large gap in the accuracy (0.610 and 0.664 of 1 in 10 R@1 for RDE and RNN, receptively). We think this is largely influenced by the data preprocessing method, because the only differences between these models is the data preprocessing, which is [58]'s contribution to the research. We are certain that our model performs better with the exquisite datasets which adapts extensive preprocessing method, because we see improvements from the RDE model to the HRDE model and additional improvements with the LTC module in all test cases (the Ubuntu-v1/v2 and the CP-QA).

In the CP-QA case, Table 3.6 indicates that the proposed RDE-LTC, HRDE, and the HRDE-LTC model show performance improvements when compared to the baseline model, TF-IDF and RDE. The average accuracy statistics are higher in the CP-QA case when compared to the Ubuntu case. We think this is due to in the smaller vocabulary size and context variety. The CP-QA dataset deals with narrower topics than in the Ubuntu dataset case. We are certain that our proposed model shows robustness in several datasets and different vocabulary size environments.

### 3.4.2 Degradation Comparison for Longer Texts

To verify the HRDE model's ability compared to the baseline model RDE, we split the testset of the Ubuntu-v1/v2 datasets based on the "number of chunks" in the $\{\text{context}\}$. Then, we measured the top-1 recall (same case as 1 in 10 R@1 in Table 3.4, and 3.5) for each group. Figure 3.5 demonstrates that the HRDE models, in darker blue and red colors, shows better performance than the RDE models, in lighter colors, for every "number of chunks" evaluations. In particular, the HRDE models are consistent when the "number-of-chunks" increased, while the RDE models degrade as the "number-of-chunks" increased.

Table 3.7: The RDE-LTC model results with different numbers of latent clusters. "Cluster 1" is the baseline model, RDE

| # clusters | Accuracy (1 in 10 R@1) | | |
|---|---|---|---|
| | Ubuntu-v1 | Ubuntu-v2 | Consumer Product QA |
| 1 | 0.643 ±0.009 | 0.610 ±0.008 | 0.869 ±0.009 |
| 2 | 0.655 ±0.005 | 0.616 ±0.006 | 0.876 ±0.011 |
| **3** | **0.656** ±0.003 | **0.625** ±0.004 | 0.877 ±0.010 |
| 4 | 0.651 ±0.005 | 0.622 ±0.005 | **0.880** ±0.009 |

### 3.4.3 Effects of the LTC Numbers

We analyze the RDE-LTC model for different numbers of latent clusters. Table 3.7 indicates that the model performances increase as the number of latent clusters increase (until 3 for the Ubuntu and 4 for the CP-QA case). This is probably a major reason for the different number of subjects in each dataset. The CP-QA dataset has an internal *category* related to the type of consumer electronic products (6 top-level *categories*; *mobile*, *office*, *photo*, *tv/video*, *accessories*, and *home appliance*), so that the LTC module makes clusters these *categories*. The Ubuntu dataset, however, has diverse contents related to issues in using the Ubuntu system. Thus, the LTC module has fewer clusters with the sparse topic compared to the CP-QA dataset.

### 3.4.4 Comprehensive Analysis of LTC

We conduct quantitative and qualitative analysis on the HRDE-LTC model for four latent topic clusters. The CP-QA dataset has *category* information; hence, latent topic clustering results can be compared with real *categories*. We randomly choose $20k$ samples containing real *category* information and evaluate each sample with the HRDE-

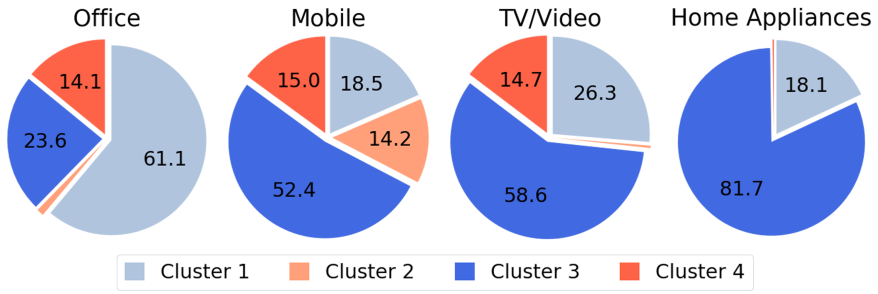



Figure 3.6: Examples of the cluster proportions for four real *categories* from $20k$ evaluated samples. Each color corresponds to each cluster.

Table 3.8: Example sentences for each cluster

| **Cluster** | **Example** |
|---|---|
| 1 | How to adjust the brightness on the s**d300 series monitors |
| 2 | How do I reject an incoming call on my Samsung Galaxy Note 3? |
| 3 | How should I clean and maintain the microwave? |
| 4 | How do I connnect my surround sound to this TV and what type of cables do I need |

LTC model. The cluster with the highest similarity among the latent topic clusters is considered a representative cluster of each sample.

Figure 3.6 shows proportion of four latent clusters among these samples according to real *category* information. Even though the HRDE-LTC model is trained without any ground-truth *category* labels, we observed that the latent cluster is formed accordingly. For instance, cluster 2 is shown mostly in "Mobile" *category* samples while "clusters 2 and 4" are rarely shown in "Home Appliance" *category* samples.

Additionally, we explore sentences with higher similarity score from the HRDE-LTC module for each four cluster. As can be seen in Table 3.8, "cluster 1" contains "screen" related sentences (e.g., brightness, pixel, display type) while "cluster 2" contains sentences with exclusive information related to the "Mobile" *category* (e.g., call

rejection, voice level). This qualitative analysis explains why "cluster 2" is shown mostly in the "Mobile" *category* in Figure 3.6. We also discover that "cluster 3" has the largest portion of samples. As "cluster 3" contains "security" and "maintenance" related sentences (e.g., password, security, log-on, maintain), we assume that this is one of the frequently asked issues across all *categories* in the CP-QA dataset. Table 3.8 shows example sentences with high scores from each cluster.

## 3.5 Further Investigation on Ranking Lengthy Document

For now, we present our investigation on ranking question-answering pairs for long text. We further investigate ranking lengthy documents in the media outlet. In particular, we rank the matching score between *headline* and *body text* in news articles [44, 76].

Misleading or false information in journalism has posed a critical social problem [77]. Much of the information shared online lacks verification and thus can put our society to unseen threats. News headlines are known to play an important role in making first impressions to readers, and thereby deciding the viral potential of news stories within social networks [78]. In digital environments under information overload, people are less likely to read or click on the whole contents but just read news headlines [79]. Likewise, much of news sharing is headline-based; people circulate news headlines without necessarily having read the full news story. On the other hand, an initial impression gained from the headline is persistent such that its stance remains even after reading the whole news content [80]. Therefore, if a news headline does not correctly represent the news story — or *is incongruent* — it could mislead readers into advocating overrated or false information, which then becomes hard to revoke.

Identifying incongruent headlines in advance will better assist readers to choose which news stories to consume, and thus decrease the chance of encountering unwanted information. Most previous research tackling this problem has tried to detect

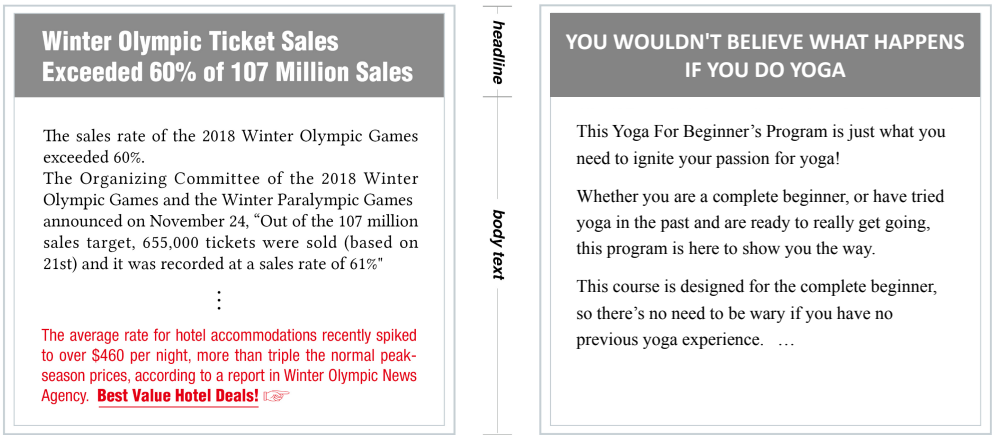



Figure 3.7: A news article with incongruent headline

incongruity in news headlines either by analyzing linguistic features of news headlines [81, 82] or by analyzing the textual similarities between news headlines and body text [54, 83]. However, lack of large-scale public dataset makes it difficult to develop sophisticated deep learning models that are better suited for such challenging detection tasks, which usually require million-scale dataset across various domains [9, 84].

### 3.5.1 Problem and Dataset

#### Problem Definition

The specific problem we tackle is **the Headline Incongruence Problem** [85], where a headline of news article holds unrelated or distinct claims with the stories across its body text. Such incongruity in news stories is a major characteristic as clickbait. Figure 3.7 demonstrates a representative example of such misinformation. The catchy news headline promises to tell certain benefits of yoga, yet the body text mainly is an advertisement for a new yoga program. Such incongruent headlines not only make a wrong impression on readers [80], but also become worse when it is shared on social media where most users just share without reading its actual contents [79]. Therefore, it is crucial to develop automated approaches that detect incongruent headlines in news

articles.

To create a new dataset, we crawled a nearly complete set of news articles published in South Korea from January of 2016 to October of 2017. From over 4 million news articles, we performed a series of cleansing steps such as removing non-critical information (e.g., reporter name, non-textual information such as photos and videos). Next, we transformed word tokens to integers, which will be released with vocab to help researchers utilize the dataset without the language barrier.

Following the above process, we generated the English-version of dataset based on a large corpus of 0.12m news articles [86]. In this paper, we do not report any results from the English version of the dataset for brevity, yet we release the two datasets together on this github page[8] for the research community.

**Training Set Creation**

It is almost impossible to manually investigate million-scale news articles for any task. Here, we automatically generated labels on crawled news articles. Rather than crafting new headlines, we implanted unrelated or topically-inconsistent content into body text of original news articles. This process can make a pair of the {headline} and {body text} where the headline tells distinct stories with its article content. Hence, the automation process for creating incongruent-labeled data involves the following steps: (1) sampling a target article from the corpora (which may be on similar topics), (2) sampling part-of-content from another article of the corpora, and (3) inserting this part-of-content to the target article.

We created congruent-labeled data by choosing them from the appropriate corpora. No single headline in this set overlaps with the incongruent-labeled data. Nonetheless, this process may incur false-negative instances when a real article having incongruent headline is chosen inappropriately as a target. We took additional steps to reduce any Type II error via rule-based pre-processing such as inspecting advertising phrases with

[8] http://github.com/david-yoon/detecting-incongruity/

an n-gram dictionary. We also hired human annotators to manually read 1,000 randomly sampled articles from the created dataset and check whether their headlines are incongruent with the article content. Above manual inspections demonstrated that our method successfully generates news articles where its headline is incongruent with its whole article content. We refer to this dataset as whole for comparison with another dataset described below.

**Paragraph Set Creation**

The task of detecting incongruent headlines can be converted into a set of sub-problems that inspect the textual relationship between a headline and each paragraph respectively, rather than examining the relationship between the headline and whole article content at once. Thus we created the paragraph dataset that transforms a pair of {headline} and {body text} into multiple sub-pairs of {headline} and {paragraph}. This conversion process not only reduces the length of text that a model should process but also increase the total number of training instances. In generating incongruent-labeled data, we sampled {headline} and {paragraph} from different articles and then matched them as pairs.

In this way, we created incongruent- and congruent-labeled datasets and maintained train, development, and test datasets that do not overlap each other. Table 3.9 shows the properties of the dataset. All datasets and their detailed description including the original news articles and the indexed version are made publicly available for the research community.

Table 3.9: Properties of the dataset. The chunk in the body text implies paragraphs and sentences for the whole and the paragraph dataset, respectively

| Dataset | # Samples | | | Headline (Avg.) | | | Body Text (Avg.) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Train | Dev. | Test | # tokens | # chunk | # tokens / chunk | # tokens | # chunk | # tokens / chunk |
| whole | 1.70M | 100,000 | 100,000 | 13.71 | 1 | 13.71 | 518.97 | 8.37 | 62.00 |
| paragraph | 14.20M | 834,064 | 100,000 | 13.71 | 1 | 13.71 | 62.00 | 2.03 | 30.05 |

### 3.5.2 Methods

Our objective is to determine whether a news article contains an incongruent headline, given a pair of {headline} and {body text}. We call the output probability being incongruent headline **incongruence score** in this paper.

#### Baseline Approaches

We introduce four baseline approaches that have been applied to the headline incongruence problem. Feature-based ensemble algorithms have been widely utilized for their simplicity and effectiveness. Among various methods, the XGBoost algorithm has shown superior performance across various prediction tasks [87]. For example, in a recent challenge on determining the stance of news articles at FNC-1, the winning team applied this algorithm based on multiple features to measure similarities between the {headline} and {body text} [88]. As a baseline, we implemented the **XGBoost (XGB)** classifier by utilizing the set of features described in the winning model, such as the cosine similarities between the {headline} and {body text}. In addition to this model, we also trained **Support Vector Machine (SVM)** classifiers based on the same set of features.

#### Recurrent Dual Encoder (RDE)

A recurrent dual encoder that is consisted of dual RNNs has been utilized to calculate a similarity between two text inputs [9]. We apply this model to the headline incongruence problem via dual RNNs that encode the {headline} and {body text}, respectively. When RNN encodes word sequences, each word is passed through a word-embedding layer that converts a word index to a corresponding 300-dimensional vector. After the encoding step, the probability of being incongruent headline is calculated by using the final hidden state of each {headline} and {body text} RNNs. The incongruence score

in the training objective is as follows:

$$
\begin{aligned}
p(\text{label}) &= \sigma((h_{t_h}^H)^\intercal M\ h_{t_b}^B + b), \\
\mathcal{L} &= -\log \prod_{n=1}^{N} p(\text{label}_n | h_{n,t_h}^H, h_{n,t_b}^B),
\end{aligned}
\tag{3.8}
$$

where $h_{t_h}^H$ and $h_{t_b}^B$ are last hidden state of each {headline} and {body text} RNN with the dimensionality $h \in \mathbb{R}^d$. The $M \in \mathbb{R}^{d \times d}$ and bias $b$ are learned model parameters. $N$ is the total number of samples used in training and $\sigma$ is the sigmoid function.

**Convolution Dual Encoder (CDE)**

Following the CNN architecture for text understanding [7], we apply Convolutional Dual Encoder to the headline incongruence problem. Taking the word sequence of {headline} and {body text} as input to the convolutional layer, we obtained a vector representation $\boldsymbol{v} = \{v_i | i = 1, \cdots, k\}$ for each part of the article through the max-over-time pooling after computing convolution with $k$ filters as follows:

$$
v_i = g(f_i(W)), \tag{3.9}
$$

where $g$ is max-over-time pooling function, $f_i$ is the CNN function with *i*-th convolutional filter, and $W \in \mathbb{R}^{t \times d}$ is a matrix of the word sequence. We use dual CNNs to encode the {headline} and the {body text} into vector representations. After encoding each part of the news article, the probability that a given article has the incongruent headline is calculated in a similar way to the equation (3.8).

**Attentive Hierarchical Dual Encoder (AHDE)**

While existing approaches perform reasonably for short text data, dealing with a long sequence of words in news articles will result in degraded performance [89, 90]. For example, the recurrent neural network utilized in RDE is poor in remembering information from the distant past. While CDE learns local dependencies between words, its typical length of convolutional filter keeps the model from capturing any relationship

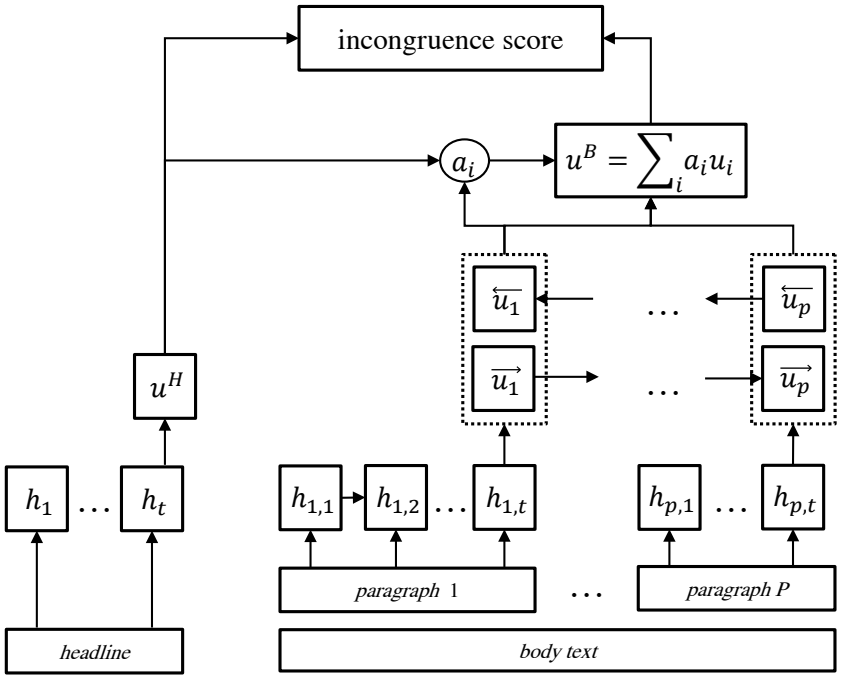



Figure 3.8: A diagram of the AHDE model. Entire text input is encoded from the word-level to the paragraph-level via employing a two-level hierarchy. The model can learn the importance of each paragraph in body text according to the headline of the article from an attention mechanism.

between the words in distinct positions. The inability to handle long sequences is a critical drawback of applying the standard deep approaches to the headline incongruence problem because a news article can be very long. The average word count in our news corpus is 518.97.

Therefore, we fill this gap by proposing neural architectures that efficiently learn hierarchical structures of long text sequences. We also present a data augmentation method that efficiently reduces the length of the target content while increasing the size of the training set.

Inspired by a previous approach that models textual similarity among question-answer pairs using a hierarchical architecture [38], this model splits text into a list of

paragraphs and encodes the entire text input from the word-level to the paragraph-level via employing a two-level hierarchy of the RNN architecture. Attention mechanism is added in paragraph-level RNN so that the model can learn the importance of each paragraph in {body text} according to {headline} of the article. Additionally, we adopt bi-directional RNNs in paragraph-level RNN to exploit information both from the past and the future.

Figure 3.8 depicts a diagram of the model. For each paragraph, the word-level RNN encodes the word sequences $\boldsymbol{w}_p = \{w_{p,1:t}\}$ to $\boldsymbol{h}_p = \{h_{p,1:t}\}$. Next, the hidden states of the word-level RNN are fed into the next-level RNN that models a sequence of paragraphs while preserving the order. The hierarchical architecture can learn textual patterns of news articles with fewer sequential steps for RNNs compared to the steps required for RDE. While RDE requires an average of 518.97 steps to learn news articles in our dataset, AHDE only accounts for 62.0 and 8.37 steps for each level of RNN, on average. The hidden states of hierarchical RNNs are as follows:

$$\begin{aligned} h_{p,t} &= f_\theta(h_{p,t-1}, w_{p,t}), \\ u_p &= g_\theta(u_{p-1}, h_p), \end{aligned} \tag{3.10}$$

where $u_p$ is the paragraph-level RNN's hidden state at the $p$-th paragraph sequence, and $h_p$ is the word-level RNN's last hidden state of each paragraph $h_p \in \{h_{1:p,t}\}$. Then, each $u_p$ of {body text} is aggregated according to its correspondence with the {headline} as follows:

$$\begin{aligned} s_p &= \mathrm{v}^\intercal tanh(W_u^B u_p^B + W_u^H u^H), \\ a_i &= \frac{\exp(s_i)}{\sum_p \exp(s_p)}, \\ u^B &= \sum_i a_i u_i^B, \end{aligned} \tag{3.11}$$

where $u_p^B$ indicates the $p$-th hidden state of the paragraph-level RNN that learns the representation of {body text}. The $u^H$ indicates the last hidden state of the paragraph-level RNN with the {headline}. We use the same training objective as the RDE model,

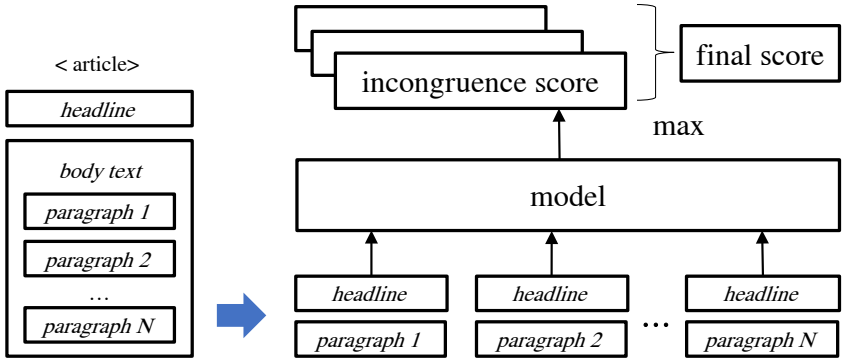



Figure 3.9: Diagram of the independent paragraph method. A given news article is split by its paragraphs, each of which is compared to the headline to calculate incongruence score. The maximum value is taken as the final incongruence score.

and the incongruence score is calculated as follows:

$$p(\text{label}) = \sigma((u^H)^\intercal M\ u^B + b) \tag{3.12}$$

**Hierarchical Recurrent Encoder (HRE)**

The AHDE model uses two hierarchical RNNs for encoding text from word-level to paragraph-level. The model requires higher computation resources in training and inference compared to those of non-hierarchical alternatives such as RDE and CDE. Therefore, we investigate an intermediate approach that models hierarchical structures of news articles with a simpler neural architecture. The text in the news {body text} is split into paragraphs, each of which is embedded by averaging the word-embedding vector from its containing words. In other words, HRE calculates $h_p$ in equation (3.10) by averaging the word embedding among the words in paragraph $p$, $h_p = \sum_i embedding(w_i),\ w_i \subset p$-th paragraph. Then, paragraph-level RNN is applied with the paragraph-encoded sequence input, $h_p$, for retrieving the final encoding vector of the whole {body text}.

**Independent Paragraph (IP) Method**

In addition to the neural architecture, we propose a data augmentation method that splits paragraphs in the {body text} and learns the relationship between each paragraph and headline independently.

Figure 3.9 depicts the diagram of the IP method, which computes **incongruence score** for each paragraph from its relationship with the news headline. The final incongruence score for the pair of {headline} and {body text} is determined as the maximum score of incongruence scores as follows:

$$p(\text{label}) = max(s_{1:p}), \tag{3.13}$$

where $s_p$ is the incongruence score calculated from the $p$-th paragraph of the {body text} and {headline}. The selection of the maximum score can better identify news articles which contain a paragraph that is highly unrelated to the news headline.

For training models with IP method, we use the paragraph dataset and incongruence scores are calculated in the following ways:

- **XGB/SVM with IP:** Extracting features from {headline} and each paragraph of {body text}, XGB/SVM measures the incongruence score for each paragraph.
- **RDE/CDE with IP:** Both models encode word sequences in each paragraph of {body text} and compare them with the encoded {headline}.
- **AHDE with IP:** To encode each paragraph in {body text}, the first-level RNN encodes word sequences for each sentence and the second-level RNN takes a sequence of sentences as input, which is retrieved from the first-level RNN.
- **HRE with IP:** To obtain the incongruence score for each paragraph, HRE first calculates the mean of word vectors for each sentence. Then, RNN encodes a sequence of sentences by taking the averaged word vectors as input.

Table 3.10: Model performance (top-2 scores marked as bold)

| Model | Without IP | | With IP | |
|---|---|---|---|---|
| | Acc. | AUROC | Acc. | AUROC |
| **SVM** | 0.640 | 0.703 | 0.677 | 0.809 |
| **XGB** | 0.677 | 0.766 | 0.729 | 0.846 |
| **CDE** | 0.812 | 0.900 | 0.870 | **0.959** |
| **RDE** | 0.845 | **0.939** | 0.863 | 0.955 |
| **AHDE** | **0.904** | **0.959** | **0.895** | **0.977** |
| **HRE** | **0.850** | 0.927 | **0.873** | 0.952 |

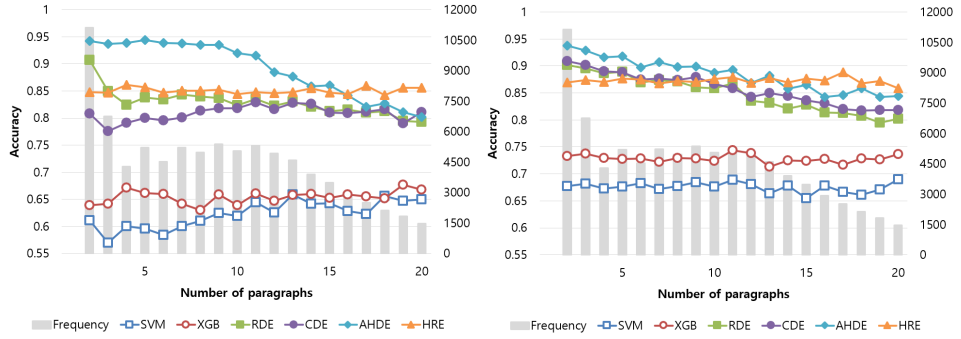
### 3.5.3 Experimental Results

In this study we use the dataset of our creation. We also use newly cralwed real world dataset for the evaluation.

#### Performance Comparison

Table 3.10 presents performances of all approaches. Performances using whole dataset is also compared with those with the IP method on paragraph dataset. We report accuracy and the AUROC (Area Under Receiver Operating Characteristic) value, which is a balanced metric with regard to the label distribution.

We first find that the four deep learning models outperform feature-based machine learning models. Among the deep learning models, the newly proposed AHDE achieved the best performance with regard to accuracy and AUROC (0.904 and 0.977, respectively). Second, prediction performance increased significantly when the IP method was applied. RDE and CDE got the advantage of the IP method most, such that they even showed the performances comparable to the hierarchical models. Even though those simple models do not have an appropriate structure to handle lengthy news data (i.e., news posts and news paragraphs on average contain 518.97 and 62.00 words respectively in Table 3.9), the IP method did help them examine the relationship between




(a) without IP method (b) with IP method

Figure 3.10: Prediction performances across a different number of paragraphs in body text of the test dataset (a) without Independent Paragraph (IP) method and (b) with IP method. Line plots demonstrate prediction accuracies with increases in the number of paragraphs, and gray bars present the frequency of the test instances having a same number of paragraphs.

the headline and each paragraph more efficiently.

### Performance over Long Text Input

One major limitation of standard deep learning approaches is difficulty in processing very long text. To verify the ability of our models in handling lengthy news articles, we measured the model performance under increasing input size. Figure 3.10 shows the result.

First, as observed in Figure 3.10(a), newly proposed models (AHDE and HRE) showed consistently higher performance over baseline approaches (e.g., XGB, RDE). When the IP data augmentation method was applied, all six models benefit and show an increase in performance as shown in Figure 3.10(b). Second, the figure suggests the robustness of our proposed models in handling long sequential input by their own hierarchical structures. Whether using the IP method or not, the newly proposed HRE and AHDE model consistently showed competitive performance irrespective of the paragraph size in {body text}. While AHDE performs better than HRE when the num-

ber of paragraphs is a few, the performance gap became narrower as paragraph size increases and HRE achieves the best score for extremely long input (i.e., a news article containing 19-20 paragraphs). This trend could be explained by the fact that HRE has a fewer number of trainable parameters compared to AHDE.

**Real World Evaluation**

To see the efficacy of our dataset and proposed models for detecting incongruent headlines in the wild, we evaluated our pre-trained models on more recent news articles. We newly gathered 232,261 news articles that were published from January to April of 2018. Testing our model with the newly gathered dataset can show the generalizability of our approach in the real world.

At first, we manually inspected random samples of news articles to see whether they have incongruent headlines. Yet, we could not retrieve enough number of articles having incongruent headlines for evaluation. This is possibly due to the sparse number of incongruent headlines in the real world. Therefore, instead of looking into the randomly sampled dataset and labeling them for evaluation, we decided to manually validate top $N$ articles by incongruence scores that are given by model prediction. Since models give incongruence scores (i.e., output probability) based on its confidence for classification, we believe such evaluation successfully estimates precision scores of prediction models. This type of evaluation is widely used in the tasks where it is impossible to count the true cases in a dataset such as question answering system [91].

Figure 3.11 shows the precision scores for AHDE models that are trained with- and without the IP method, respectively. The x-axis presents the top-N articles by incongruence scores that are retrieved by the models out of the newly gathered articles over 4 months. The y-axis demonstrates the precision value corresponding to the top N articles.

Here we make three observations. First, the AHDE model with the IP augmenta-

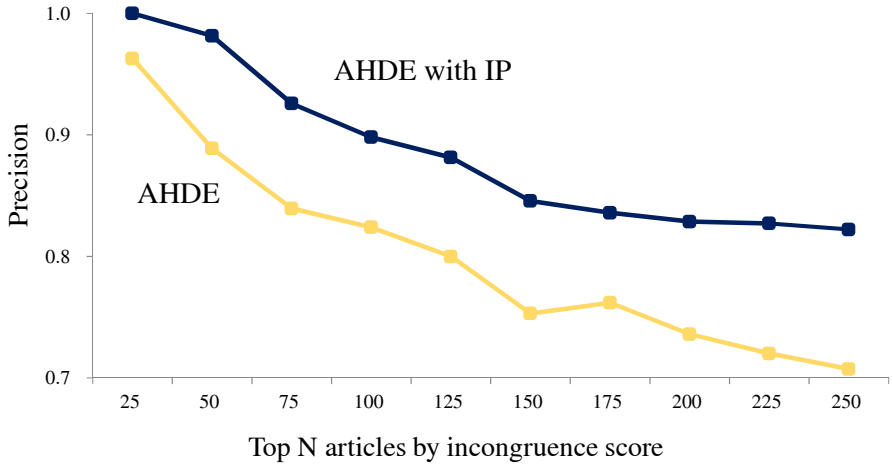



Figure 3.11: Precision values for detecting news articles with incongruent headlines in the newly gathered dataset. The x-axis shows the top-N articles by incongruence scores, and the y-axis presents its corresponding precision.

tion consistently shows higher precision than the AHDE model without the IP method. This finding supports the superior performance of the IP method across different evaluations. Second, the AHDE model with IP achieved the precision of 1.0 for the top 25 articles. Even though the model was trained by a separate dataset, it successfully filtered out real cases where its headline conveys different stories with associated body text. Third, when we evaluate the top 250 articles, the precision of the AHDE model with IP reduced to 0.82. Nevertheless, this precision value is high enough to be utilized for detection in real news platforms.

The above observations suggest that our approach and dataset could be applied for the headline incongruence problem in the wild. Based on application scenarios, one could control the trade-off between precision and recall by changing the model thresholds.

## 3.6 Conclusion

In this study, we proposed the HRDE model and LTC module. HRDE showed higher performances in ranking answer candidates and less performance degradations when dealing with longer texts compared to conventional models. The LTC module provided additional performance improvements when combined with both RDE and HRDE models, as it added latent topic cluster information according to dataset properties. With this proposed model, we achieved state-of-the-art performances in Ubuntu datasets. We also evaluated our model in real world question answering dataset, Consumer Product QA. This demonstrated the robustness of the proposed model with the best results.

We further investigate ranking lenghy document comprise of *headline* and *body text*. We enhance HRDE models by employing the attention mechanism and show the proposed neural network efficiently learn the textual relationship between headline and body text via a hierarchical recurrent architecture. The experiments demonstrate that the models trained on our released corpus show decent performances both on the synthetic dataset and on the real-world dataset.

# Chapter 4

# Answer-Selection for Short Sentence

Automatic question answering (QA) is a primary objective of artificial intelligence. Recently, research on this task has taken two major directions based on the answer span considered by the model. The first direction (i.e., the fine-grained approach) finds an exact answer to a question within a given passage [14]. The second direction (i.e., the coarse-level approach) is an information retrieval (IR)-based approach that provides the most relevant sentence from a given document in response to a question. In this study, we are interested in building a model that computes a matching score between two text inputs. In particular, our model is designed to undertake an answer-selection task that chooses the sentence that is most relevant to the question from a list of answer candidates. This task has been extensively investigated by researchers because it is a fundamental task that can be applied to other QA-related tasks [1, 2, 3, 4, 30, 31].

However, most previous answer-selection studies have employed small datasets [26, 55] compared with the large datasets employed for other natural language processing (NLP) tasks [9, 14]. Therefore, the exploration of sophisticated deep learning models for this task is difficult. Furthermore, we find that some previous research do not report performance on both development and test dataset but only note one of them. It raises uncertainty in reproducing the results and in applying to other tasks.

To fill this gap, we conduct an intensive investigation with the following directions

to obtain the best performance in the answer-selection task [92]. First, we explore the effect of additional information by adopting a pretrained language model (**LM**) to compute the vector representation of the input text. Recent studies have shown that replacing the word-embedding layer with a pretrained language model helps the model capture the contextual meaning of words in the sentence [6, 24]. Following this study, we select an ELMo [6] language model for this study. We investigate the applicability of transfer learning (**TL**) using a large-scale corpus that is created for a relevant-sentence-selection task (i.e., question-answering NLI (QNLI) dataset [52]). Second, we further enhance one of the baseline models, **Comp-Clip** [2] (refer to the discussion in 4.2.1), for the target QA task by proposing a novel latent clustering (**LC**) method. The **LC** method computes latent cluster information for target samples by creating a latent memory space and calculating the similarity between the sample and the memory. By an end-to-end learning process with the answer-selection task, the **LC** method assigns *true*-label question-answer pairs to similar clusters. In this manner, a model will have further information for matching sentence pairs, which increases the total model performance. Last, we explore the effect of different objective functions (listwise and pointwise learning). In contrast to previous research [2], we observe that the pointwise learning approach performs better than the listwise learning approach when we apply our proposed methods. Extensive experiments are conducted to investigate the efficacy and properties of the proposed methods and show the superiority of our proposed approaches for achieving state-of-the-art performance with the WikiQA and TREC-QA datasets.

## 4.1 Related Work

Researchers have investigated models based on neural networks for question-answering tasks. One study employs a Siamese architecture that utilizes an encoder (e.g., RNN or CNN) to compute vector representations of the question and the answer. The affinity

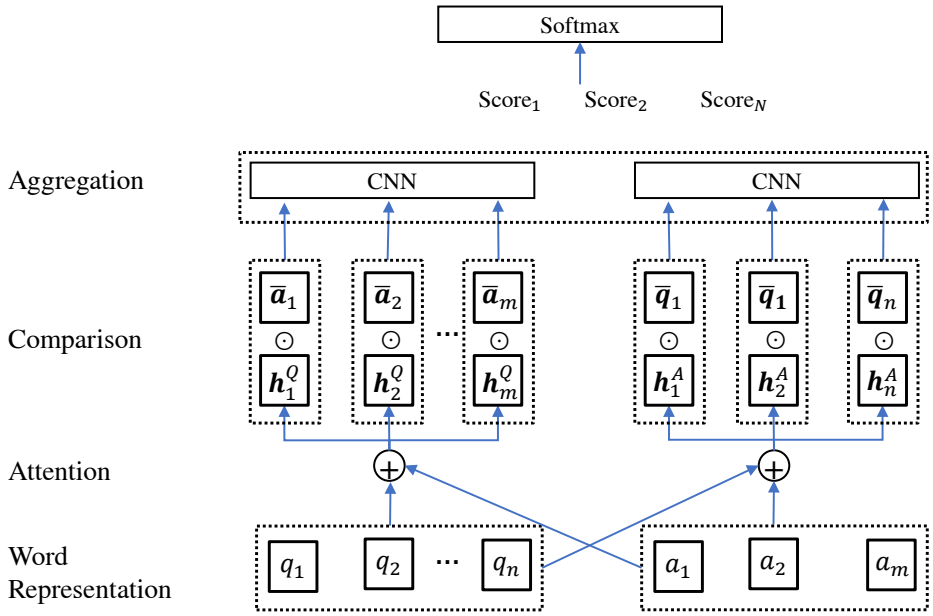



Figure 4.1: Architecture of the baseline model, Compare-Aggregate Model with Dynamic-Clip Attention [2].

score is calculated based on these vector representations [9]. To improve the model performance by enabling the use of information from one sentence (e.g., a question or an answer) in computing the representation of another sentence, researchers included the attention mechanism in their models [54, 59, 93].

Another line of research includes the compare-aggregate framework [1]. In this framework, first, vector representations of each sentence are computed. Second, these representations are compared. Last, the results are aggregated to calculate the matching score between the question and the answer [2, 3, 4]. Figure 4.1 shows the baseline model, Compare-Aggregate Model with Dynamic-Clip Attention, that we improve in this research.

In this study, unlike the previous research, we employ a pretrained language model and a latent-cluster method to help the model understand the information in the question and the answer.

## 4.2 Method

### 4.2.1 Baseline approach

In this paper, we are interested in estimating the matching score $f(y|\mathbf{Q},\mathbf{A})$, where $y$, $\mathbf{Q}=\{q_1,...,q_n\}$ and $\mathbf{A}=\{a_1,...,a_m\}$ represent the label, the question and the answer, respectfully. We select the **Comp-Clip Model** from [2], which is referred to as the **Comp-Clip** model, as our baseline model. The model consists of the following four parts:

#### Context representation

The question $\mathbf{Q}\in\mathbb{R}^{d\times Q}$ and answer $\mathbf{A}\in\mathbb{R}^{d\times A}$, (where $d$ is a dimensionality of word embedding and Q and A are the length of the sequence in $\mathbf{Q}$ and $\mathbf{A}$, respectively), are processed to capture the contextual information and the word as follows:

$$\begin{aligned}\overline{\mathbf{Q}} &= \sigma(\mathbf{W}^i\mathbf{Q})\odot\tanh(\mathbf{W}^u\mathbf{Q}),\\ \overline{\mathbf{A}} &= \sigma(\mathbf{W}^i\mathbf{A})\odot\tanh(\mathbf{W}^u\mathbf{A}),\end{aligned}\tag{4.1}$$

where $\odot$ denotes element-wise multiplication, and $\sigma$ is the sigmoid function. The $\mathbf{W}\in\mathbb{R}^{l\times d}$ is the learned model parameter.

#### Attention

The soft alignment of each element in $\overline{\mathbf{Q}}\in\mathbb{R}^{l\times Q}$ and $\overline{\mathbf{A}}\in\mathbb{R}^{l\times A}$ are calculated using dynamic-clip attention [2]. We obtain the corresponding vectors $\mathbf{H}^Q\in\mathbb{R}^{l\times A}$ and $\mathbf{H}^A\in\mathbb{R}^{l\times Q}$.

$$\begin{aligned}\mathbf{H}^Q &= \overline{\mathbf{Q}}\cdot\ \text{softmax}((\mathbf{W}^q\overline{\mathbf{Q}})^\intercal\overline{\mathbf{A}}),\\ \mathbf{H}^A &= \overline{\mathbf{A}}\cdot\ \text{softmax}((\mathbf{W}^a\overline{\mathbf{A}})^\intercal\overline{\mathbf{Q}}).\end{aligned}\tag{4.2}$$

### Comparison

A comparison function is used to match each word in the question and answer to a corresponding attention-applied vector representation:

$$\begin{aligned} \mathbf{C}^Q &= \overline{\mathbf{A}} \odot \mathbf{H}^Q, \;\; (\mathbf{C}^Q \in \mathbb{R}^{l \times A}), \\ \mathbf{C}^A &= \overline{\mathbf{Q}} \odot \mathbf{H}^A, \;\; (\mathbf{C}^A \in \mathbb{R}^{l \times Q}), \end{aligned} \tag{4.3}$$

where $\odot$ denotes element-wise multiplication.

### Aggregation

We aggregate the vectors from the comparison layer using CNN [7] with $n$-types of filters and calculate the matching score between **Q** and **A**.

$$\begin{aligned} &\mathbf{R}^Q = \text{CNN}(\mathbf{C}^Q), \;\; \mathbf{R}^A = \text{CNN}(\mathbf{C}^A), \\ &\text{score} = \sigma([\mathbf{R}^Q; \mathbf{R}^A]^\intercal \, \mathbf{W}), \end{aligned} \tag{4.4}$$

where [;] denotes concatenation of each vector $\mathbf{R}^Q \in \mathbb{R}^{nl}$ and $\mathbf{R}^A \in \mathbb{R}^{nl}$. The $\mathbf{W} \in \mathbb{R}^{2nl \times 1}$ is the learned model parameter.

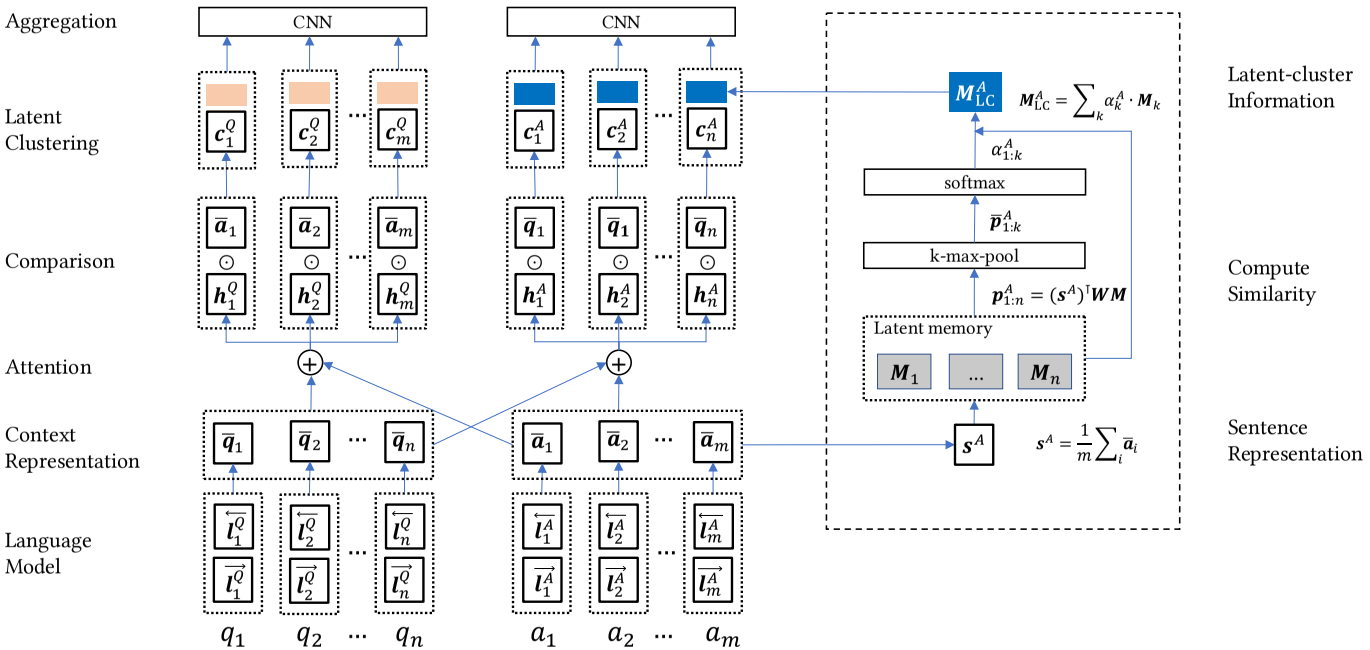



Figure 4.2: The architecture of the model. The dotted box on the right shows the process through which the latent-cluster information is computed and added to the answer. This process is also performed in the question part but is omitted in the figure. The latent memory is shared in both processes.

### 4.2.2 Proposed Approaches (Comp-Clip+LM+LC+TL)

To achieve the best performance in the answer-selection task, we propose four approaches: adding a pretrained **LM**; adding the **LC** information of each sentence as auxiliary knowledge; applying **TL** to benefit from large-scale data; and modifying the objective function from listwise to pointwise learning. Figure 4.2 depicts the total architecture of the proposed model.

#### Pretrained Language Model (LM)

In previous research, each word in the question or answer is converted to a vector representation using a word-embedding layer that is randomly initialized or transferred from an existing pretrained word-embedding model [5, 49].

Recent studies have shown that replacing the word embedding layer with a pretrained **LM** helps the model capture the contextual meaning of the words in the sentence [6, 24]. We select an ELMo [6] language model and replace the previous word embedding layer with the ELMo model as follows: $\mathbf{L}^{\mathrm{Q}} = \mathrm{ELMo}(\mathbf{Q}),\ \mathbf{L}^{\mathrm{A}} = \mathrm{ELMo}(\mathbf{A})$. These new representations—$\mathbf{L}^{\mathrm{Q}}$ and $\mathbf{L}^{\mathrm{A}}$—are substituted for $\mathbf{Q}$ and $\mathbf{A}$, respectively, in equation (4.1).

#### Latent Clustering (LC) Method

We assume that extracting the **LC** information of the text and using it as auxiliary information will help the neural network model analyze the corpus. The dotted box in Figure 4.2 shows the proposed **LC** method. We create $n$-many latent memory vectors $\mathbf{M}_{1:n}$ and calculate the similarity between the sentence representation and each latent memory vector. The latent-cluster information of the sentence representation will be obtained using a weighted sum of the latent memory vectors according to the calcu-

Passage:
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail… Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

Question: What causes precipitation to fall?

Answer: gravity

Figure 4.3: An example of the SQuAD dataset.

lated similarity as follows:

$$\begin{aligned}
\mathbf{p}_{1:n} &= \mathbf{s}^{\intercal}\mathbf{W}\mathbf{M}_{1:n}, \\
\overline{\mathbf{p}}_{1:k} &= k\text{-max-pool}(\mathbf{p}_{1:n}), \\
\alpha_{1:k} &= \text{softmax}(\overline{\mathbf{p}}_{1:k}), \\
\mathbf{M}_{\text{LC}} &= \sum_{k} \overline{\alpha}_k \mathbf{M}_k,
\end{aligned} \tag{4.5}$$

where $\mathbf{s} \in \mathbb{R}^d$ is a sentence representation, $\mathbf{M}_{1:n} \in \mathbb{R}^{d' \times n}$ indicates the latent memory, and $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is the learned model parameter.

We apply the **LC** method and extract cluster information from each question and answer. This additional information is added to each of the final representations in the comparison part (see 4.2.1) as follows:

$$\begin{aligned}
\mathbf{M}^Q_{\text{LC}} &= f(\frac{\sum_i \overline{q}_i}{n}),\ \overline{q}_i \subset \overline{\mathbf{Q}}_{1:n}, \\
\mathbf{M}^A_{\text{LC}} &= f(\frac{\sum_i \overline{a}_i)}{m},\ \overline{a}_i \subset \overline{\mathbf{A}}_{1:m}, \\
\mathbf{C}^Q_{\text{new}} &= [\mathbf{C}^Q; \mathbf{M}^Q_{\text{LC}}],\ \ \mathbf{C}^A_{\text{new}} = [\mathbf{C}^A; \mathbf{M}^A_{\text{LC}}],
\end{aligned} \tag{4.6}$$

where $f$ is the **LC** method (in equation 4.5) and [;] denotes the concatenation of each vector. These new representations—$\mathbf{C}^Q_{\text{new}}$ and $\mathbf{C}^A_{\text{new}}$—are substituted for $\mathbf{C}^Q$ and $\mathbf{C}^A$ in

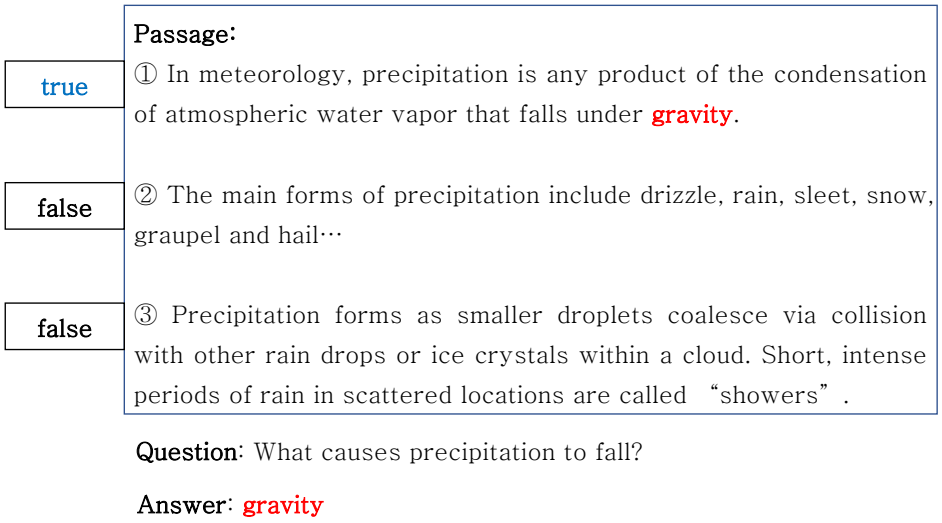



Figure 4.4: A process to generate QNLI dataset from SQuAD dataset.

equation (4.4). Note that we average word-embedding to obtain sentence representation in the previous equation.

### Transfer Learning (TL)

To observe the efficacy in a large dataset, we apply transfer learning using the question-answering NLI (QNLI) corpus [52].

Figure 4.3 shows an example of the SQuAD dataset [14]. The dataset comprises of *passage*, *question*, and *answer*. The object of SQuAD is find the answer span in the passage given question. The QNLI dataset is generated from the SQuAD dataset. First, all the sentences are split from the passage; then each of them is paired with the question. Second, the *true*-label is given to each pair that contains *answer* word/phrase in the sentence. Otherwise, the *false*-label is assigned to each pair that does not contain the answer word/phrase in it as shown in Figure 4.4. In this way, a large amount of {question,sentence,label} triples can be generated.

We train the **CompClip** model with the QNLI corpus and then fine-tune the model with target corpora, such as the WikiQA [55] and TREC-QA [26] datasets.

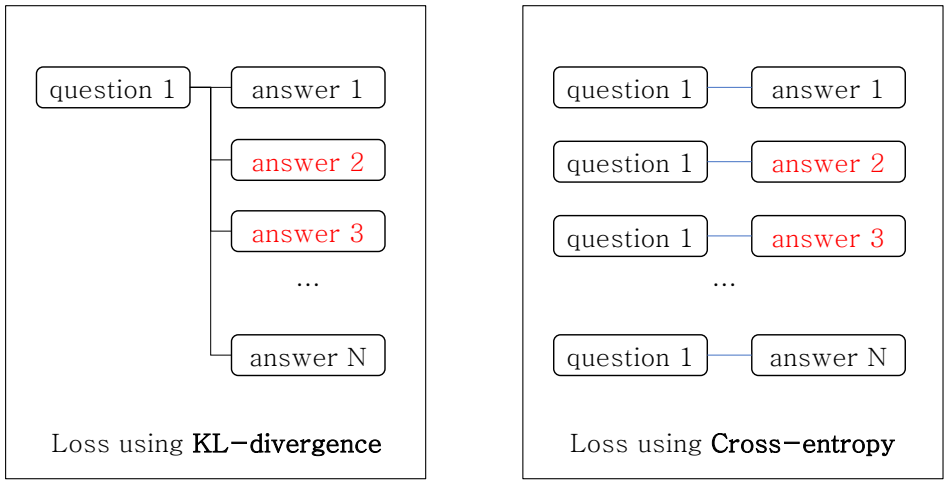



Figure 4.5: Differences between list-wise and pair-wise loss.

**Pointwise Learning to Rank**

Previous research adopts a listwise learning approach. With a dataset that consists of a question, $\mathbf{Q}$, a related answer set, $\mathbf{A} = \{\mathbf{A}_1, ..., \mathbf{A}_N\}$, and a target label, $\mathbf{y} = \{y_1, ..., y_N\}$, a matching score is computed using equation (4.4). This approach applies KL-divergence loss to train the model as follows:

$$\begin{aligned} \text{score}_i &= \text{model}(\mathbf{Q}, \mathbf{A}_i), \\ \mathbf{S} &= \text{softmax}([\text{score}_1, ..., \text{score}_i]), \\ \text{loss} &= \sum_{n=1}^{N} \text{KL}(\mathbf{S}_n || \mathbf{y}_n), \end{aligned} \tag{4.7}$$

where $i$ is the number of answer candidates for the given question and $N$ is the total number of samples employed during training (see the left side in Figure 4.5).

In contrast, we pair each answer candidate to the question and compute the cross-entropy loss to train the model as follows:

$$\text{loss} = -\sum_{n=1}^{N} y_n \log\left(\text{score}_n\right), \tag{4.8}$$

where $N$ is the total number of samples used during training (see the right side in

Table 4.1: Properties of the dataset

| Dataset | Listwise pairs | | | Pointwise pairs | | |
|---|---|---|---|---|---|---|
| | train | dev | test | train | dev | test |
| WikiQA | 873 | 126 | 243 | 8.6k | 1.1k | 2.3k |
| TREC-QA | 1.2k | 65 | 68 | 53k | 1.1k | 1.4k |
| QNLI | 86k | 10k | - | 428k | 169k | - |

Figure 4.5). Using this approach, the number of training instances for a single iteration increases, as shown in Table 4.1.

## 4.3 Experimental Setup and Dataset

We regard all tasks as relevant answer selections for the given questions. Following the previous study, we report the model performance as the mean average precision (MAP) and the mean reciprocal rank (MRR)[1]. Figure 4.6, 4.7 show how MAP and MRR compute the matching score, repectively.

To test the performance of the model, we utilize the TREC-QA, WikiQA and QNLI datasets [26, 55, 52].

### 4.3.1 Dataset

**WikiQA** [55] is an answer selection QA dataset constructed from real queries of Bing and Wikipedia. Following the literature [2, 3], we use only questions that contain at least one correct answer among the list of answer candidates. There are 873/126/243 questions and 8,627/1,130/2,351 question-answer pairs for train/dev/test split.

**TREC-QA** [26] is another answer selection QA dataset created from the TREC Question-Answering tracks. In this study, we use the clean dataset that removed questions from

[1] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

Table 4.2: Example of the TREC-QA dataset

| **Question** |
|---|
| "what was the monetary value of the nobel peace prize in 1989 ?" |

| **Answer** | *flag* |
|---|---|
| "each nobel prize is worth $ 469,000 ." | 1 |
| "the nobel prize in physics was awarded today to two americans and a west german whose work led to the atomic clock used as an international standard ." | 0 |
| "the chemistry prize went to two americans for the discovery of surprising properties of the genetic material rna ." | 0 |
| "the nobel prize in chemistry is shared by thomas cech , 41 , of the university of colorado , and sidney altman , 50 , of yale university ." | 0 |
| "half the physics prize will go to ramsey , 74 ." | 0 |
| "americans have shared or won the chemistry prize 36 times among the 112 times it has been awarded since 1901 ." | 0 |
| "fifty-two of the 134 recipients of the physics prize have been americans ." | 0 |

the dev and test datasets that did not have answers or had only positive/negative answers. There are 1,229/65/68 questions and 53,417/1,117/1,442 question-answer pairs for train/dev/test split. Table 4.2 shows an example of the TREC-QA dataset.

**QNLI** [52] is a modified version of the SQuAD dataset [14] that allows for sentence selection QA. The context paragraph in SQuAD is split into sentences, and each sentence is paired with the question. The true label is given to the question-sentence pairs when the sentence contains the answer (see Figure 4.4). There are 86,308/10,385 questions and 428,998/169,435 question-answer pairs for train/dev split. Considering the large size of this dataset, we use it to train the base model for transfer learning; it is

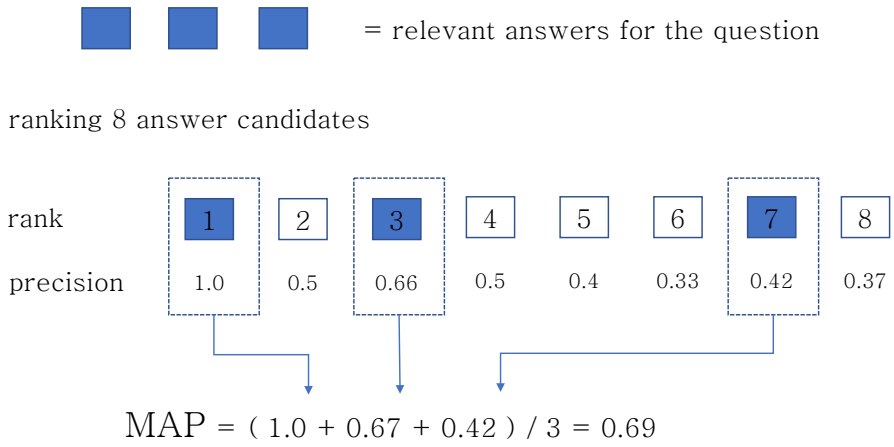



Figure 4.6: The MAP measures average precision values at ranks of relevant answers.

also used to evaluate the proposed model performance in a large dataset environment.

### 4.3.2 Implementation Details

To implement the **Comp-Clip** model, we apply a context projection weight matrix with 100 dimensions that are shared between the question part and the answer part (eq. 4.1). In the aggregation part, we use 1-D CNN with a total of 500 filters, which involves five types of filters $K \in \mathbb{R}^{\{1,2,3,4,5\} \times 100}$, 100 per type. The weight matrices for the filters were initialized using the Xavier method [94]. This CNN is independently applied to the question part and answer part. For the **LC** method, we perform additional hyper-parameter searching experiments to select the best parameters. We select $k$ (for the $k$-max-pool in equation 4.5) as 6 and 4 for the WikiQA and TREC-QA case, respectively. In both datasets, we apply 8 latent clusters.

The vocabulary size in the WiKiQA, TREC-QA and QNLI dataset are 30,104, 56,908 and 154,442, respectively. When applying the **TL**, the vocabulary size is set to 154,442, and the dimension of the context projection weight matrix is set to 300. We use the Adam optimizer, including gradient clipping, by the norm at a threshold of 5. For the purpose of regularization, we applied a dropout with a ratio of 0.5.

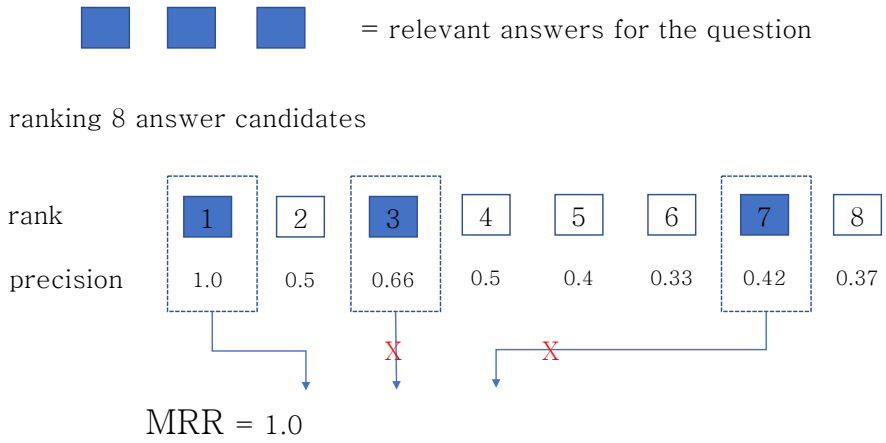



Figure 4.7: The MRR average precision values of the first relevant answers.

## 4.4 Empirical Results

### 4.4.1 Comparison with Other Methods

Table 4.3 shows the model performance for the WikiQA and TREC-QA datasets. For the Compare-Aggregate (2016), Comp-Clip (2017), IWAN (2017) and IWAN+sCARNN (2018) models, we measure the performance on the WikiQA dataset using the authors' implementations (marked by * in the table). Unlike previous studies, we report our results for both the dev dataset and the test dataset because we note a performance gap between these datasets. While training the model, we apply an early stop that is based on the performance of the dev dataset and measure the performance on the test dataset. Because **Comp-Clip** [2] is our baseline model, we implement it from scratch and achieve a performance that is similar to that of the original paper.

**WikiQA:** For the WikiQA dataset, the pointwise learning approach shows a better performance than the listwise learning approach. We combine **LM** with the base model (**Comp-Clip +LM**) and observe a significant improvement in performance in terms of MAP (0.714 to 0.746 absolute). When we add the **LC** method (**Comp-Clip +LM +LC**), the best previous results are surpassed in terms of MAP (0.718 to 0.764 abso-

lute). We achieve a vast improvement in performance in terms of the MAP (0.764 to 0.834 absolute) by including the **TL** approach (**Comp-Clip + LM + LC + TL**).

**TREC-QA:** The pointwise learning approach also shows excellent performance with the TREC-QA dataset. As shown in Table 4.1, the TREC-QA dataset has a larger number of answer candidates per question. We assume that this characteristic prevents the model from handling the dataset with a listwise learning approach. As in the WikiQA case, we achieve additional performance gains in terms of the MAP as we apply **LM**, **LC**, and **TL** (0.850, 0.868 and 0.875, respectively). In particular, our model outperforms the best previous result when we add **LC** method, (**Comp-Clip +LM +LC**) in terms of MAP (0.865 to 0.868).

Table 4.3: Model performance (the top 3 scores are marked in bold for each task). We evaluate model [1, 2, 3, 4] on the WikiQA corpus using author's implementation (marked by *). For TREC-QA case, we present reported results in the original papers.

| **Model** | **WikiQA** | | | | **TREC-QA** | | | |
|---|---|---|---|---|---|---|---|---|
| | MAP | | MRR | | MAP | | MRR | |
| | dev | test | dev | test | dev | test | dev | test |
| Compare-Aggregate (2017) [1] | 0.743* | 0.699* | 0.754* | 0.708* | - | - | - | - |
| Comp-Clip (2017) [2] | 0.732* | 0.718* | 0.738* | 0.732* | - | 0.821 | - | 0.899 |
| IWAN (2017) [3] | 0.738* | 0.692* | 0.749* | 0.705* | - | 0.822 | - | 0.899 |
| IWAN + sCARNN (2018) [4] | 0.719* | 0.716* | 0.729* | 0.722* | - | 0.829 | - | 0.875 |
| MCAN (2018) [30] | - | - | - | - | - | 0.838 | - | **0.904** |
| Question Classification (2018) [31] | - | - | - | - | - | **0.865** | - | 0.904 |
| **Listwise Learning to Rank** | | | | | | | | |
| Comp-Clip (our implementation) | 0.756 | 0.708 | 0.766 | 0.725 | 0.750 | 0.744 | 0.805 | 0.791 |
| Comp-Clip (our implementation) + LM | 0.783 | 0.748 | 0.791 | 0.768 | 0.825 | 0.823 | 0.870 | 0.868 |
| Comp-Clip (our implementation) + LM + LC | 0.787 | 0.759 | 0.793 | 0.772 | 0.841 | 0.832 | 0.877 | 0.880 |
| Comp-Clip (our implementation) + LM + LC +TL | 0.822 | **0.830** | 0.836 | **0.841** | 0.866 | 0.848 | 0.911 | 0.902 |
| **Pointwise Learning to Rank** | | | | | | | | |
| Comp-Clip (our implementation) | 0.776 | 0.714 | 0.784 | 0.732 | 0.866 | 0.835 | 0.933 | 0.877 |
| Comp-Clip (our implementation) + LM | 0.785 | 0.746 | 0.789 | 0.762 | 0.872 | 0.850 | 0.930 | 0.898 |
| Comp-Clip (our implementation) + LM + LC | 0.782 | **0.764** | 0.785 | **0.784** | 0.879 | **0.868** | 0.942 | **0.928** |
| Comp-Clip (our implementation) + LM + LC +TL | 0.842 | **0.834** | 0.845 | **0.848** | 0.913 | **0.875** | 0.977 | **0.940** |

Table 4.4: Model (Comp-Clip +LM +LC) performance on the QNLI corpus with a variant number of clusters (top score marked as bold)

| # Clusters | Listwise Learning | | Pointwise Learning | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| 1 | 0.822 | 0.819 | 0.842 | 0.841 |
| 4 | 0.839 | 0.840 | 0.846 | 0.845 |
| 8 | **0.841** | **0.842** | 0.846 | **0.846** |
| 16 | 0.840 | **0.842** | **0.847** | **0.846** |

### 4.4.2 Impact of Latent Clustering

To evaluate the impact of latent clustering method (**Comp-Clip +LM +LC**) in a larger dataset environment, we perform QNLI evaluation. Table 4.4 shows the performance of the model (**Comp-Clip +LM +LC**) for the QNLI dataset with a variant number of clusters. Note that the QNLI dataset is created from the SQuAD [14] dataset, which only provides train and dev subsets. Consequently, we report the model performances for the dev dataset. As shown in the table, we achieve the best results with 8 clusters in listwise learning and 16 clusters in pointwise learning. In both cases, we achieve no additional performance gain after 16 clusters.

## 4.5 Conclusion

In this study, our proposed method achieves state-of-the-art performance for both the WikiQA dataset and TREC-QA dataset. We show that leveraging a large amount of data is crucial for capturing the contextual representation of input text. In addition, we show that the proposed latent clustering method with a pointwise objective function significantly improves the model performance in the sentence-level QA task.

# Chapter 5

# Supporting Sentence Detection for Question Answering

Understanding texts and being able to answer a question posed by a human is a long-standing goal in the artificial intelligence field. With the rapid advancement of neural network-based models and the availability of large-scale datasets, such as SQuAD [14] and TriviaQA [16], researchers have begun to concentrate on building automatic question-answering (QA) systems. One example of such a system is the machine-reading question-answering (MRQA) model, which provides answers to questions from given passages [66, 21, 22, 95].

Recently, research has revealed that most questions in existing MRQA datasets do not require reasoning across sentences in the given context (passage); instead, they can be answered by looking at only a single sentence [96]. Using this characteristic, a simple model can achieve performance competitive with that of a sophisticated model. Furthermore, the existing MRQA models can be distracted in the inference stage by adversarial sentences inserted into the original passage, which causes severe performance degradation [97]. However, in most real scenarios of QA applications, more than one sentence should be utilized to extract a correct answer.

To alleviate this limitation of previous datasets, another type of dataset was developed in which answering the question requires reasoning over multiple sentences in the given passages [17, 20]. Figure 5.1 shows an example of a recently released

**Passage 1**, 2015 Diamond Head Classic:

① *The 2015 Diamond Head Classic was a mid-season eight-team college basketball tournament…* ② *It was the seventh annual Diamond Head Classic tournament …* ③ *No. 3-ranked Oklahoma defeated Harvard to win the tournament championship...* ④ *Buddy Hield was named the tournament's MVP.*

⋮

**Passage N**, Buddy Hield:

① *Chavano Rainier "Buddy" Hield is a Bahamian professional basketball player for the Sacramento Kings of the NBA...* ② …

**Question**: Which team does the player named 2015 Diamond Head Classic's MVP play for?

**Supporting Sentences**: **1**-④, **N**-①

Figure 5.1: An example of dataset. Detecting *supporting sentences* is an essential step being able to answer the question.

dataset, the HotpotQA. This dataset consists of not only question-answer pairs with context passages but also *supporting sentence* information for answering the question annotated by a human.

In this study, we build a model that exploits the relational information among sentences in passages to classify the *supporting sentences* that contain the essential information for answering the question. To this end, we propose a novel graph neural network model named **propagate-selector (PS)**, which can be directly employed as a subsystem in the QA pipeline. First, we design a graph structure to hold information in the HotpotQA dataset by assigning each sentence to an independent graph node. Then, we connect the undirected edges between nodes using a proposed graph topology (see the discussion in the "Proposed Method" section). Next, we allow **PS** to propagate information between the nodes through iterative hops to perform reasoning across the given sentences. Through the propagation process, the model learns to understand in-

formation that cannot be inferred when considering sentences in isolation.

This work is the first to employ a graph neural network structure to find *supporting sentences* for a QA system. Through experiments, we demonstrate that compared with the widely used answer-selection models [1, 2, 3, 4, 92], the proposed method achieves better performance when classifying *supporting sentences*.

## 5.1 Related Work

Previous researchers have also investigated neural network-based models for MRQA. One line of inquiry employs an attention mechanism between tokens in the question and passage to compute the answer span from the given text [21, 22]. As the task scope was extended from specific- to open-domain QA, several models have been proposed to select a relevant paragraph from the text to predict the answer span [98, 99]. However, none of these methods have addressed reasoning over multiple sentences.

To understand the relational patterns in the dataset, graph neural network algorithms have also been proposed. [100] proposed a graph convolutional network to classify graph-structured data. This model was further investigated for applications involving large-scale graphs [101], for the effectiveness of aggregating and combining graph nodes by employing an attention mechanism [102], and for adopting recurrent node updates [103]. These methods successfully demonstrated their potential and effectiveness in understanding relational datasets, such as entity linking in heterogeneous knowledge graphs, product recommendation systems, and detecting side effects in drug [104, 105, 106]. In addition, one trial involved applying graph neural networks to QA tasks; however, this usage was limited to the entity level rather than sentence-level understanding [107].

## 5.2 Method

Our objective in this study is to identify *supporting sentences*, among sentences in the given text that contain information essential for answering the question. To tackle this problem, we first introduce answer-selection models, which are widely studied in the research community. These models are considered strong baselines since they can be directly applied to our task with the same objective function. Then we describe our proposed method.

### 5.2.1 Baseline approaches

We introduce baseline models for the answer-selection task, which have been extensively studied and have proved their efficacy to the research community. These models are developed to compute the matching similarity between any pairs of text (the question and the target sentence in our case).

**Compare Aggregate Framework (CompAggr).**

This model [1] computes matching similarity between two texts (the question and the target sentence). It consists of attention, comparison, and aggregation parts.

**Attention:** The soft alignment of the question $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and target sentence $\mathbf{S} \in \mathbb{R}^{d \times S}$ (where $d$ is a dimensionality of word embedding and $Q$ and $S$ are the length of the sequences in the question and sentence, respectively) is computed by applying an attention mechanism over the column vector in $\mathbf{Q}$ for each column vector in $\mathbf{S}$. With the computed alignment, we obtain a corresponding vector $\mathbf{A}^Q \in \mathbb{R}^{d \times S}$ as follows:

$$\mathbf{A}^Q = \mathbf{Q} \cdot \text{softmax}((\mathbf{WQ})^\intercal \mathbf{S}), \tag{5.1}$$

where $\mathbf{W}$ is a learned model parameter matrix.

**Comparison:** An element-wise multiplication is employed as a comparison function

to combine each pair of $\mathbf{A}^Q$ and $\mathbf{S}$ into a vector $\mathbf{C} \in \mathbb{R}^{d \times S}$.

**Aggregation:** [7]'s CNN with $n$-types of filters is applied to aggregate all information in the vector $\mathbf{C}$. Finally, the model employs a fully connected layer to compute the matching score between the question and the target sentence as follows:

$$\begin{aligned} \mathbf{R} &= \text{CNN}(\mathbf{C}), \;\; (\mathbf{R} \in \mathbb{R}^{nd}), \\ \hat{y}_c &= \text{softmax}((\mathbf{R})^\intercal \, \mathbf{W} + \mathbf{b} \,), \end{aligned} \tag{5.2}$$

where $\hat{y}_c$ is the predicted probability for the target class, $c$, and $\mathbf{W} \in \mathbb{R}^{nd \times c}$ and bias $\mathbf{b}$ are learned model parameters.

The loss function for the model is cross-entropy between predicted labels and true-labels as follows:

$$\mathcal{L} = -\log \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}), \tag{5.3}$$

where $y_{i,c}$ is the true label vector and $\hat{y}_{i,c}$ is the predicted probability from the softmax layer. $C$ is the total number of classes (true and false for this task), and $N$ is the total number of samples used in training.

**CompAggr-kMax.**

This model [2] is an extension of the **CompAggr** model. The only differences lie in applying attention (in equation (5.1)) to both the $\mathbf{Q}$ and $\mathbf{S}$ side and applying k-max polling after the softmax function as follows:

$$\begin{aligned} \mathbf{A}^Q &= \mathbf{Q} \cdot \text{softmax}((\mathbf{WQ})^\intercal \mathbf{S}), \\ \mathbf{A}^S &= \mathbf{S} \cdot \text{softmax}((\mathbf{WS})^\intercal \mathbf{Q}). \end{aligned} \tag{5.4}$$

**CompClip-LM-LC.**

This model [92] is an extension of the **CompAggr-kMax** model. It employs the ELMo [6] model to enhance the word embedding layer for the question and target sentence by adopting the pretrained contextual language model. Additionally, it develops a latent

clustering method to automatically compute topic information in texts and to use it as auxiliary information to improve the model performance.

**IWAN.**

This model [3] is a variation model based on the compare aggregate framework. Unlike **CompAggr**, it employs RNNs to encode a sequence of the words in the text (question and target sentence independently). At the same time, it computes an inter-alignment weight between the question and the target sentence. The matching score is computed by aggregating this information (question, target sentence, and inter-aligned representation).

**sCARNN.**

This model [4] is an extension of the **IWAN** model. It proposes a novel recurrent unit to regulate the flow of the input (sequence of words in a text) and then replaces the RNNs in the **IWAN** model of the proposed unit.

### 5.2.2 Proposed Approach (Propagate-Selector)

To build a model that can perform reasoning across multiple sentences, we propose a graph neural network model called **Propagate-selector** (**PS**). **PS** consists of four parts as follows (topology, node representation, aggregation, and update):

**Topology**

The topology of the graph determines the connections among the nodes in the graph. These connections will be used as a path that allows the information to flow from one node to another. Figure 5.2 depicts the topology of the proposed model. In an offline step, we organize the content of each instance in a graph, where each node represents a sentence from the passages and the question. Then, we add edges between nodes using the following topology:

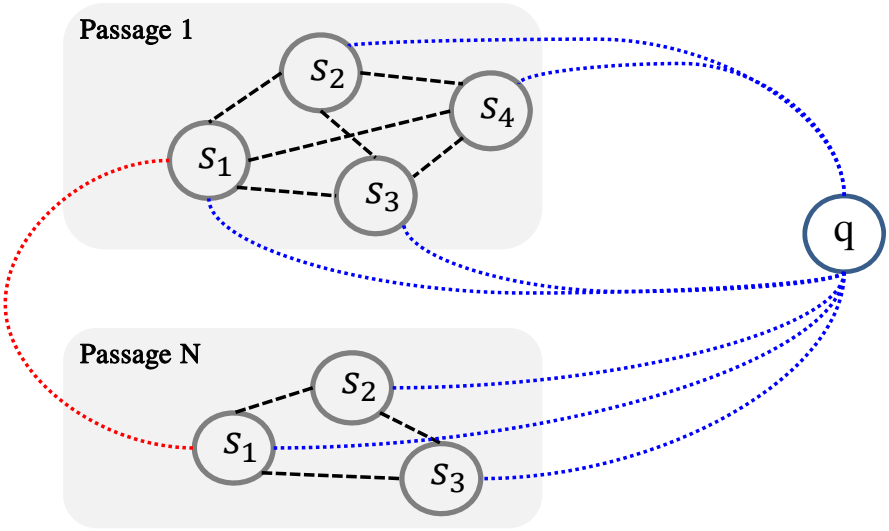



Figure 5.2: Topology of the proposed model. Each node represents a sentence from the passage and the question.

- we fully connect nodes that represent sentences from the same passage (dotted-black);
- we fully connect nodes that represent the first sentence of each passage (dotted-red);
- we add an edge between the question and every node for each passage (dotted-blue).

In this way, we enable a path by which sentence nodes can propagate information between both inner and outer passages. Furthermore, we investigate different strategies for connecting those nodes in the graph and examine their corresponding effectiveness for detecting *supporting sentences* in the text (see the discussion in "Impact of Various Graph Topologies" section).

**Node representation**

Question $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and sentence $\mathbf{S}_i \in \mathbb{R}^{d \times S_i}$ (where $d$ is the dimensionality of the word embedding and $Q$ and $S_i$ represent the lengths of the sequences in $\mathbf{Q}$ and $\mathbf{S}_i$, respectively) are processed to acquire the sentence-level information. Recent studies have shown that a pretrained language model helps the model capture the contextual meaning of words in the sentence [6, 24]. Following this study, we select an ELMo [6] language model for the word-embedding layer of our model as follows:

$$\begin{aligned} \mathbf{L}^Q &= \text{ELMo}(\mathbf{Q}),\ (\mathbf{L}^Q \in \mathbb{R}^{d \times Q}), \\ \mathbf{L}^S &= \text{ELMo}(\mathbf{S}),\ (\mathbf{L}^S \in \mathbb{R}^{d \times S}). \end{aligned} \tag{5.5}$$

Using these new representations, we compute the sentence representation as follows:

$$\begin{aligned} \mathbf{h}_t^Q &= f_\theta(\mathbf{h}_{t-1}^Q, \mathbf{L}_t^Q), \\ \mathbf{h}_t^S &= f_\theta(\mathbf{h}_{t-1}^S, \mathbf{L}_t^S), \\ \mathbf{N}^Q &= \mathbf{h}_{\text{last}}^Q,\ \ \mathbf{N}^S = \mathbf{h}_{\text{last}}^S, \end{aligned} \tag{5.6}$$

where $f_\theta$ is the RNN function with the weight parameters $\theta$ and $\mathbf{N}^Q \in \mathbb{R}^{d'}$ and $\mathbf{N}^S \in \mathbb{R}^{d'}$ are node representations for the question and sentence, respectively (where $d'$ is the dimensionality of the RNN hidden units).

As computing the node representation is an essential process for acquiring information from a text, we investigate various approaches for encoding sentences, such as replacing the ELMo word representations using different methods (the GloVe [49] or the BERT [24]) and replacing the RNN function in equation (5.6) with the pooling method. Furthermore, we adopt the universal sentence encoding method based on the recently developed transformer model [108]. Detailed information will be given in the "Impact of Node Representation" section.

**Aggregation**

An iterative attentive aggregation function to the neighbor nodes is utilized to compute the amount of information to be propagated to each node in the graph as follows:

$$\begin{aligned}
\mathbf{A}_v^{(k)} &= \sigma(\sum_{u \in N(v)} a_{vu}^{(k)}\ \mathbf{W}^{(k)} \cdot\ \mathbf{N}_u^{(k)}), \\
a_{vu}^{(k)} &= \frac{\exp(\mathbf{S}_{vu})}{\sum_k \exp(\mathbf{S}_{vk})}, \\
\mathbf{S}_{vu}^{(k)} &= (\mathbf{N}_v^{(k)})^\intercal \cdot \mathbf{W}^{(k)} \cdot\ \mathbf{N}_u^{(k)},
\end{aligned} \tag{5.7}$$

where $\mathbf{A}_v \in \mathbb{R}^{d'}$ is the aggregated information for the *v*-th node computed by attentive weighted summation of its neighbor nodes, $a_{vu}$ is the attention weight between node *v* and its neighbor nodes $u$ ($u \in N(v)$), $\mathbf{N}_u \in \mathbb{R}^{d'}$ is the *u*-th node representation, $\sigma$ is a nonlinear activation function, and $\mathbf{W} \in \mathbb{R}^{d' \times d'}$ is the learned model parameter. Because all the nodes belong to a graph structure in which the iterative aggregation is performed among nodes, the *k* in the equation indicates that the computation occurs in the *k*-th hop (iteration).

**Update**

The aggregated information for the *v*-th node, $\mathbf{A}_v$ in equation (5.7), is combined with its previous node representation to update the node. We apply a skip connection to allow the model to learn the amount of information to be updated in each hop as follows:

$$\mathbf{N}_v^{(k)} = \sigma(\mathbf{W} \cdot \{\mathbf{N}_v^{(k-1)}; \mathbf{A}_v^{(k)}\}), \tag{5.8}$$

where $\sigma$ is a nonlinear activation function, $\{;\}$ indicates vector concatenation, and $\mathbf{W} \in \mathbb{R}^{d' \times 2d'}$ is the learned model parameter.

**Optimization**

Because our objective is to classify *supporting sentences* ($S_i \in P_n$) from the given tuples $<Q, P_n, Y_i>$, we define two types of loss to be minimized. One is a rank loss

that computes the cross-entropy loss between a question and each sentence using the ground-truth $Y_i$ as follows:

$$\begin{aligned} &\text{loss}_{rank} = -\log \sum_{i=1}^{N} Y_i \log(S_i), \\ &\text{S} = [\text{score}_1, ..., \text{score}_i], \\ &\text{score}_i = g_\theta(\mathbf{N}^Q, \mathbf{N}_i^S), \end{aligned} \tag{5.9}$$

where $g_\theta$ is a feedforward network that computes a similarity score between the final representation of the question and each sentence. The other is attention loss, which is defined in each hop as follows:

$$\text{loss}_{attn} = -\log \sum_{i=1}^{k} \sum_{i=1}^{N} Y_i \log(a_{qi}^{(k)}), \tag{5.10}$$

where $a_{qi}^{(k)}$ indicates the relevance between the question node *q* and the *i*-th sentence node in the *k*-th hop as computed by equation (5.7).

Finally, these two losses are combined to construct the final objective function:

$$\mathcal{L} = \alpha \, \text{loss}_{rank} + \text{loss}_{attn}, \tag{5.11}$$

where $\alpha$ is a hyperparameter.

## 5.3 Experimental Setup and Dataset

### 5.3.1 Dataset

The specific problem we aim to tackle in this study is to classify *supporting sentences* in the MRQA task. We consider the target dataset HotpotQA, by [17], which comprises tuples (<$Q$, $P_n$, $Y_i$, $A$>) in which $Q$ is the question, $P_n$ is the set of passages as the given context, and each passage $P \in P_n$ further comprises a set of sentences $S_i$ ($S_i \in P_n$). Here, $Y_i$ is a binary label indicating whether $S_i$ contains the information required to answer the question, and $A$ is the answer. In particular, we call a sentence,

Table 5.1: Properties of the HotpotQA dataset

| **properties** | **train** | **dev** |
|---|---|---|
| # questions | 90,447 | 7,405 |
| # sentences | 3,703,344 | 306,487 |
| passages / question | 9.95 | 9.95 |
| sentences / passage | 4.12 | 4.16 |
| sentences / question | 40.94 | 41.39 |
| supporting sentences / question | 2.39 | 2.43 |
| avg tokens (question) | 17.92 | 15.83 |
| avg tokens (sentence) | 22.38 | 22.41 |

$S_s \in S_i$, a *supporting sentence* when $Y_s$ is *true*. Figure 5.1 shows an example of the HotpotQA dataset.

In this study, we do not use the answer information from the dataset; we use only the subsequent tuples $<Q, P_n, Y_i>$ when classifying *supporting sentences*. We believe that this subproblem plays an important role in building a full QA pipeline because the proposed models for this task will be combined with other MRQA models in an end-to-end training process.

Similar to the answer-selection task in the QA literature, we report the model performance using the mean average precision (MAP) and mean reciprocal rank (MRR) metrics. Table 5.1 shows the properties of the dataset. We conduct a series of experiments to compare baseline methods with the newly proposed models.

### 5.3.2 Implementation Details

To implement the **propagate-selector** (**PS**) model, we first use a small version of ELMo (13.6 *M* parameters) that provides 256-dimensional context embedding. This choice was based on the available batch size (50 for our experiments) when training the complete model on a single GPU (GTX 1080 Ti). When we tried using the origi-

Table 5.2: Model performance on the HotpotQA dataset (top scores marked in bold)

| **Model** | dev | | train | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| **IWAN** [3] | 0.526 | 0.680 | 0.605 | 0.775 |
| **sCARNN** [4] | 0.534 | 0.698 | 0.620 | 0.792 |
| **CompAggr** [1] | 0.659 | 0.812 | 0.796 | 0.911 |
| **CompAggr-kMax** [2] | 0.670 | 0.825 | 0.767 | 0.901 |
| **CompClip-LM-LC** [92] | 0.702 | 0.848 | 0.757 | 0.884 |
| **PS**-*rnn-elmo-s* | 0.716 | 0.841 | 0.813 | 0.916 |
| **PS**-*rnn-elmo* | **0.734** | **0.853** | **0.863** | **0.945** |

nal version of ELMo (93.6 *M* parameters, 1024-dimensional context embedding), we were able to increase the batch size only up to 20, which results in excessive training time (approximately 90 hours). For the sentence encoding, we used a GRU [51] with a hidden unit dimension of 200. The hidden unit weight matrix of the GRU is initialized using orthogonal weights [73]. Dropout [75] is applied for regularization purposes at a ratio of 0.7 for the RNN (in equation (5.6)) to 0.7 for the attention weight matrix (in equation (5.7)). For the nonlinear activation function (in equation (5.7) and (5.8)), we use the $tanh$ function.

Regarding the vocabulary, we replaced vocabulary with fewer than 12 instances in terms of term-frequency with "*UNK*" tokens. The final vocabulary size was 138,156. We also applied the Adam optimizer [74], including gradient clipping by norm at a threshold of 5.

Table 5.3: Model performance with original (5.5B) version of ELMo

| # hop | dev | | train | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| 1 | 0.651 | 0.794 | 0.716 | 0.842 |
| 2 | 0.653 | 0.797 | 0.721 | 0.850 |
| 3 | 0.698 | 0.830 | 0.800 | 0.908 |
| **4** | **0.734** | **0.853** | **0.863** | **0.945** |
| 5 | 0.700 | 0.827 | 0.803 | 0.906 |
| 6 | 0.457 | 0.606 | 0.467 | 0.621 |

## 5.4 Empirical Results

### 5.4.1 Comparisons with Other Methods

Table 5.2 shows the model performances on the HotpotQA dataset. Because the dataset only provides training (trainset) and validation (devset) subsets, we report the model performances on these datasets. While training the model, we implement early termination based on the devset performance and measure the best performance. To compare the model performances, we choose widely used answer-selection models such as **IWAN** [3], **sCARNN** [4], **CompAggr** [1], **CompAggr-kMax** [2], and **CompClip-LM-LC** [92], which were primarily developed to rank candidate answers for a given question (refer to the "Baseline approaches" section for detailed information on the models). In addition to the main proposed model, **PS**-*rnn-elmo*, we also report the model performance with a small version of ELMo, **PS**-*rnn-elmo-s*.

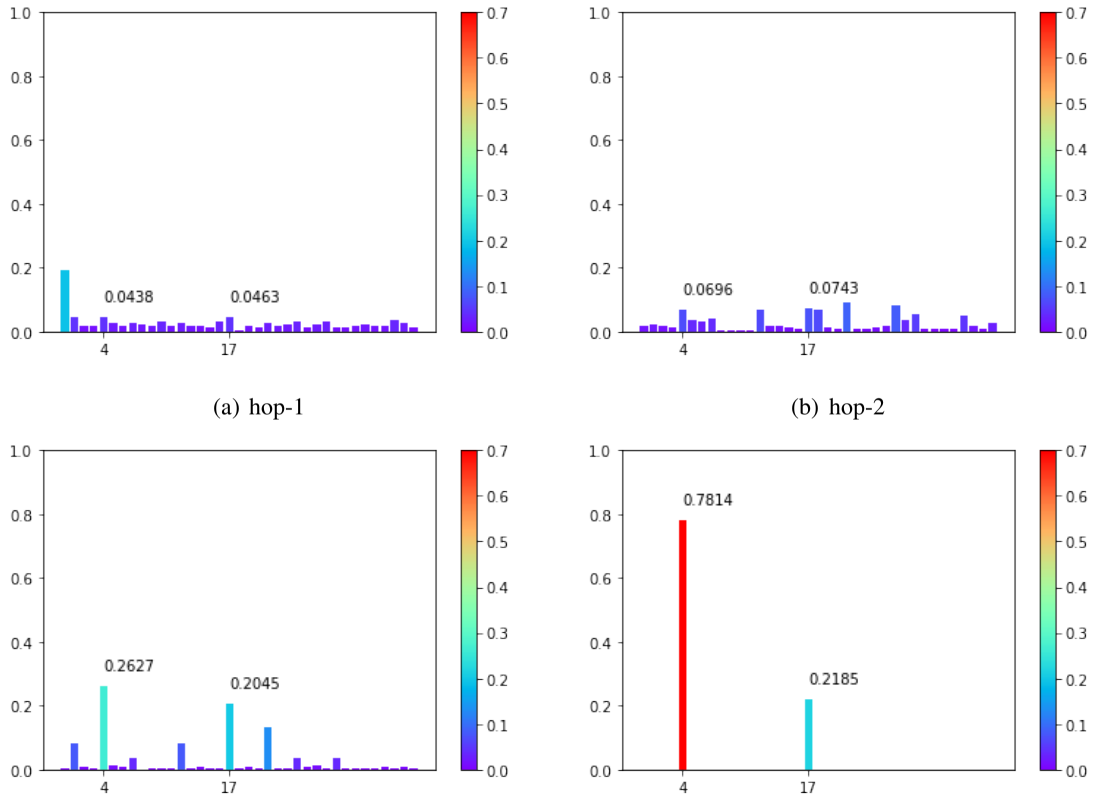
As shown in Table 5.2, the proposed **PS**-*rnn-elmo* shows a significant MAP performance improvement compared to the previous best model, **CompClip-LM-LC** (0.702 to 0.734 absolute).

Table 5.4: Model performance with small version of ELMo

| # hop | dev | | train | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| 1 | 0.648 | 0.790 | 0.708 | 0.842 |
| 2 | 0.655 | 0.801 | 0.720 | 0.853 |
| 3 | 0.681 | 0.816 | 0.768 | 0.886 |
| 4 | 0.706 | 0.834 | 0.796 | 0.906 |
| **5** | **0.716** | **0.841** | **0.813** | **0.916** |
| 6 | 0.441 | 0.596 | 0.452 | 0.600 |

### 5.4.2 Hop Analysis

Table 5.3 shows the model performance (**PS**-*rnn-elmo*) as the number of hops increases. We find that the model achieves the best performance in the 4-hop case but starts to degrade when the number of hops exceeds 4. We assume that the model experiences the vanishing gradient problem under a larger number of iterative propagations (hops). Table 5.4 shows the model performance with the small version of ELMo as the number of hop increases.




(a) hop-1 (b) hop-2

(c) hop-3 (d) hop-4

Figure 5.3: Attention weights between the question and sentences in the passages. As the number of hops increases, the proposed model correctly classifies *supporting sentences* (ground-truth index 4 and 17).

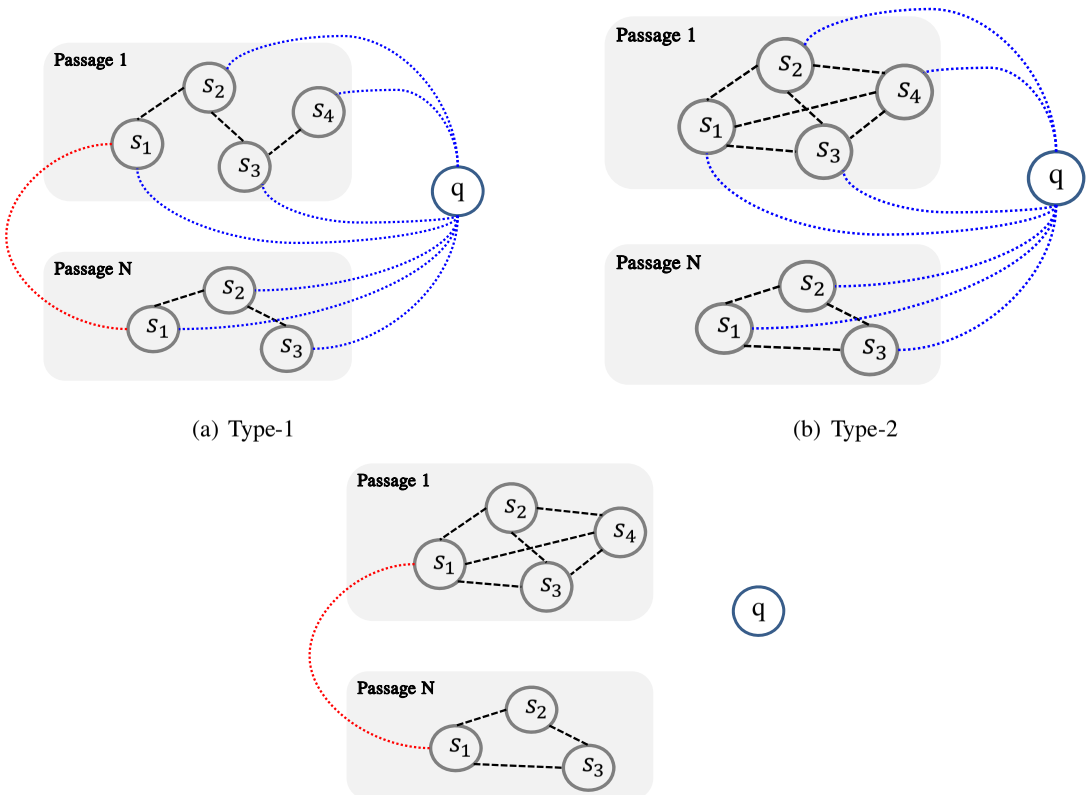
Table 5.5: Model performance with different typologies. The connection strategies between nodes for each type are illustrated in Figure 4.4.

| **Model** | dev | | train | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| **PS**-*rnn-elmo-s* | **0.716** | **0.841** | **0.813** | **0.916** |
| **Type-1** (*rnn-elmo-s*) | 0.694 | 0.834 | 0.807 | 0.915 |
| **Type-2** (*rnn-elmo-s*) | 0.705 | 0.836 | 0.792 | 0.903 |
| **Type-3** (*rnn-elmo-s*) | 0.658 | 0.796 | 0.729 | 0.857 |

Figure 5.3 depicts the attention weight between the question node and each sentence node (hop-4 model case). As the hop number increases, we observe that the model properly identifies *supporting sentences* (in this example, sentences #4 and #17). This behavior demonstrates that our proposed model correctly learns how to propagate the necessary information among the sentence nodes via the iterative process.

### 5.4.3 Impact of Various Graph Topologies

The topology of the graph determines the path by which information flows and is aggregated. To see the quantitative contributions of each connection in the graph, we perform ablation experiments as follows:




(a) Type-1

(b) Type-2

(c) Type-3

Figure 5.4: Different typologies for the graph. Type-1 reduce connection within the passage, type-2 remove connection between the passages and type-3 remove connection between each sentence node and the question node.

Table 5.6: Model performance with the different method for computing node representation.

| **Model** | dev | | train | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| **PS**-*USD_T* | 0.651 | 0.795 | 0.693 | 0.830 |
| **PS**-*avg-glove* | 0.617 | 0.753 | 0.876 | 0.945 |
| **PS**-*avg-elmo-s* | 0.471 | 0.611 | 0.483 | 0.625 |
| **PS**-*rnn-glove* | 0.700 | 0.822 | **0.919** | **0.971** |
| **PS**-*rnn-elmo-s* | 0.716 | 0.841 | 0.813 | 0.916 |
| **PS**-*rnn-elmo* | **0.734** | **0.853** | 0.863 | 0.945 |
| **PS**-*rnn-bert* | 0.667 | 0.806 | 0.708 | 0.841 |

- **Type-1**: We reduce the connections between sentences within the same passage. Only the previous and next sentences are connected to their neighbor sentence (see Figure 5.4(a)).
- **Type-2**: We remove the connections between the passages, which reveals the contribution of the information flow among independent passages (see figure 5.4(b)).
- **Type-3**: We remove the connections between each sentence node and the question node (see figure 5.4(c)).

Figure 5.4 illustrates different types of connection strategies, and Table 5.5 shows their corresponding performances. To reach the best performance, we conduct experiments multiple times by changing the number of hops in the model from 1 to 6 for each case (Type-1 to Type-3). From the experiment, hop-4 is selected as the best-performing hyper-parameter. However, all the model variations undergo performance degradation compared to the original topology (**PS**-*rnn-elmo-s*).

### 5.4.4 Impact of Node Representation

To see the effectiveness of the various approaches for computing sentence representation, we investigate combinations of well-studied methods.

#### Word Representation

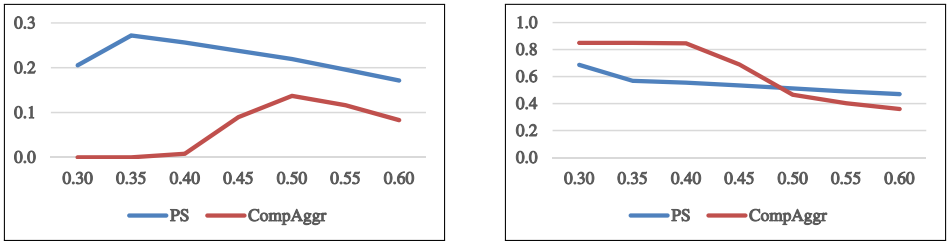
Vector representations of the words in each sentence are computed from the original version of ELMo model (-elmo), small version of ELMo model (-elmo-s), BERT [24] model (-bert), or mapped to GloVe word embedding (-glove).

#### Node Representation

Each node representation is computed by employing three general methods for encoding the sequence of word representations as follows:

- We employ an RNN model (-rnn) to encode sequential information in the sentence. The final representation of the RNN's hidden status is considered as a node representation (see equation (5.6)).
- We apply a pooling method (-avg) that averages all the word representations in the sentence to compute the node representation as follows: $\mathbf{N}^Q = \text{average}(\mathbf{Q})$, $\mathbf{N}^S = \text{average}(\mathbf{S})$. These new representations-$\mathbf{N}^Q$ and $\mathbf{N}^S$-are substituted for the node representations in equation (5.6).
- We adopt the pretrained universal sentence-encoding method (-USD_T), which is based on the recently developed transformer model [108]. This model computes sentence representation directly from the sequence of words in any text.

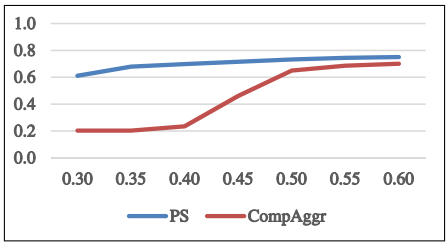
Table 5.6 depicts the model performance with different node representation methods. In all cases, the RNN encoding skims (-rnn) performs better than that of the average pooling (-avg). Interestingly, average pooling with ELMo representation (**PS**-*avg-elmo-s*) performs worse than in the GloVe representation (**PS**-*avg-glove*) case. From



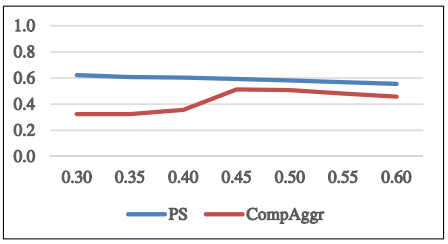

(a) exact match




(b) recall




(c) precision




(d) f1

Figure 5.5: Model performance with various measure. The x-axis shows a threshold value that is used for determining the label of the question-supporting sentence pair by the confidence score.

this result, we find that averaging ELMo does not produce proper node representations. For the **PS**-*rnn-bert* case, we do not fine-tune the BERT model and only use its computing word representation. We expect there exists a possibility to enhance model performance by fine-tuning the BERT with the end-to-end training process.

## 5.5 Discussion

In this study, we focus on a model that can detect *supporting sentences* to answer a question. We do not consider competition against the full QA systems, i.e., machine reading QA (MRQA) models, which are jointly trained with two objectives, "*extracting answer span*" and "*detecting supporting sentence*". Note that we do not use the exact "answer span" information when detecting the *supporting sentences*. We think "answer-span" supervision allows the model to track the *supporting sentences* from

simple word matching. Therefore, our investigations are focused on evaluating and analyzing the effectiveness of the proposed graph neural network-based model for classifying *supporting sentences* compared to the well-known answer-selection QA models. To evaluate the performance of the proposed model from a different perspective, we adopt other traditional measures for the QA system (i.e., precision, recall, and f1), and evaluate our methods. These measures require a specific label (*true* or *false*) for each pair of data (the question and the supporting sentence candidate). As our model computes the confidence scores for each pair of data, we give a *true*-label when the confidence scores are greater than a predefined threshold value (otherwise, we give the pair a *false*-label). Figure 5.5 shows the model performances (**PS**-*rnn-elmo* vs **CompAggr**) in regards to the variation of the threshold value (0.3 to 0.6).

In future research directions, we will investigate the best way to combine our proposed model with existing MRQA algorithms to build a full QA system. It would also be possible to link the current graph to another graph (i.e., knowledge graph) to engage external knowledge information in the question-answering system. We also hope that our work inspires future works aiming to perform multihop reasoning on free-form text.

## 5.6 Conclusion

In this study, we propose a graph neural network that finds the sentences crucial for answering a question. The experiments demonstrate that the model correctly classifies *supporting sentences* by iteratively propagating the necessary information through its novel architecture. We believe that our approach will play an important role in building a QA pipeline in combination with other MRQA models trained in an end-to-end manner.

# Chapter 6

# Conclusion

In this dissertation, we propose novel ranking algorithms for the various QA systems based on deep neural networks. We first tackle the long-text QA that requires the model to understand the excessively large sequence of text inputs. To tackle this challenge, we propose a hierarchical recurrent dual encoder model that encodes passage to a vector representation from a word-level to a chunk-level to effectively capture the entire meaning. By adapting the hierarchical structure, the HRDE shows very small performance degradations in lengthy text comprehension while other state-of-the-art recurrent neural network models suffer from it. We further propose a latent topic clustering method that allowing each data sample to find its nearest topic cluster, thus helping the neural network model to analyze the entire data.

Secondly, we investigate the short-text QA, where the information in text pairs are limited. To overcome the insufficiency, we combine a pretrained language model and an enhanced latent clustering method to the QA model. This novel architecture enables the model to utilizes additional information, resulting in achieving state-of-the-art performance for the standard answer-selection tasks (i.e., WikiQA, TREC-QA).

Finally, we investigate detecting supporting sentences for complex QA system. This system requires a model to analyze relationships behind each sentence in a passage to answer the question. Inspired by the hierarchical nature of the text, we propose

a graph neural network-based model that presents each sentence as nodes and sentential structure as their corresponding edges. By iterative information propagation process between connected nodes, the proposed model understands the relationship in the passage and be able to classify the necessary sentence to answer the question correctly.

# Bibliography

[1] Shuohang Wang and Jing Jiang, "A compare-aggregate model for matching text sequences," in *International Conference on Learning Representations (ICLR)*, 2017.

[2] Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin, "A compare-aggregate model with dynamic-clip attention for answer selection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1987–1990.

[3] Gehui Shen, Yunlun Yang, and Zhi-Hong Deng, "Inter-weighted alignment network for sentence pair modeling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1179–1189.

[4] Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui, and Hung Bui, "The context-dependent additive recurrent neural net," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, vol. 1, pp. 1274–1283.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *International Conference on Learning Representations (ICLR) Workshop*, 2013.

[6] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 2227–2237.

[7] Yoon Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[9] Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," in *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015, p. 285.

[10] Philipp Cimiano, Christina Unger, and John McCrae, "Ontology-based interpretation of natural language," *Synthesis Lectures on Human Language Technologies*, vol. 7, no. 2, pp. 1–178, 2014.

[11] Zhiping Zheng, "Answerbus question answering system," in *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., 2002, pp. 399–404.

[12] James Fan, Aditya Kalyanpur, DC Gondek, and David A Ferrucci, "Automatic knowledge extraction from documents," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 5–1, 2012.

[13] Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty, "Finding needles in the haystack: Search and candidate generation," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 6–1, 2012.

[14] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[15] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng, "Ms marco: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.

[16] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 1601–1611.

[17] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380.

[18] Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette, "The narrativeqa reading comprehension challenge," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 317–328, 2018.

[19] Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot, "Wikireading: A novel

large-scale language understanding task over wikipedia," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1535–1545.

[20] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel, "Constructing datasets for multi-hop reading comprehension across documents," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 287–302, 2018.

[21] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, "Bidirectional attention flow for machine comprehension," in *International Conference on Learning Representations (ICLR)*, 2016.

[22] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou, "Gated self-matching networks for reading comprehension and question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 189–198.

[23] Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang, "Multi-passage machine reading comprehension with cross-passage answer verification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 1918–1927.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[25] Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, and Hai Zhao, "Sg-net: Syntax-guided machine reading comprehension," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[26] Mengqiu Wang, Noah A Smith, and Teruko Mitamura, "What is the jeopardy model? a quasi-synchronous grammar for qa," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[27] Tomasz Jurczyk, Michael Zhai, and Jinho D Choi, "Selqa: A new benchmark for selection-based question answering," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 820–827.

[28] Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor, "Semeval-2017 task 3: Community question answering," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 27–48.

[29] Lishuang Li, Anqiao Zhou, Beibei Zhang, and Fengsen Xiao, "Multiple fragment-level interactive networks for answer selection," *Neurocomputing*, 2020.

[30] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui, "Multi-cast attention networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2299–2308.

[31] Harish Tayyar Madabushi, Mark Lee, and John Barnden, "Integrating question classification and deep learning for improved answer selection," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 3283–3294.

[32] Sanjay Kamath, Brigitte Grau, and Yue Ma, "Predicting and integrating expected answer types into a simple recurrent neural network model for answer sentence selection," *Computación y Sistemas*, vol. 23, no. 3, 2019.

[33] Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin, "Bridging the gap between relevance matching and semantic matching for short text

similarity modeling," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5373–5384.

[34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[35] Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara, "A gated self-attention memory network for answer selection," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5955–5961.

[36] Siddhant Garg, Thuy Vu, and Alessandro Moschitti, "Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[37] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al., "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[38] Seunghyun Yoon, Joongbo Shin, and Kyomin Jung, "Learning to rank question-answer pairs using hierarchical recurrent encoder with latent topic clustering," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1575–1584.

[39] Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu, "Modeling multi-turn conversation with deep utterance aggregation," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 3740–3752.

[40] Junyu Lu, Chenbin Zhang, Zeying Xie, Guang Ling, Tom Chao Zhou, and Zenglin Xu, "Constructing interpretive spatio-temporal features for multi-turn responses selection," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 44–50.

[41] Taesun Whang, Dongyub Lee, Chanhee Lee, Kisu Yang, Dongsuk Oh, and HeuiSeok Lim, "Domain adaptive training bert for response selection," *arXiv preprint arXiv:1908.04812*, 2019.

[42] Jia-Chen Gu, Zhen-Hua Ling, and Quan Liu, "Utterance-to-utterance interactive matching network for multi-turn response selection in retrieval-based chatbots," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 369–379, 2019.

[43] Jia-Chen Gu, Zhen-Hua Ling, and Quan Liu, "Interactive matching network for multi-turn response selection in retrieval-based chatbots," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2321–2324.

[44] Seunghyun Yoon, Kunwoo Park, Joongbo Shin, Hongjun Lim, Seungpil Won, Meeyoung Cha, and Kyomin Jung, "Detecting incongruity between news headline and body text via a deep hierarchical encoder," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 791–800.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[46] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[47] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[48] John R Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in linguistic analysis*, 1957.

[49] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[50] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *NIPS Deep Learning Workshop*, 2014.

[52] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018, pp. 353–355.

[53] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association of Computational Linguistics*, vol. 4, no. 1, pp. 259–272, 2016.

[54] Zhiguo Wang, Wael Hamza, and Radu Florian, "Bilateral multi-perspective matching for natural language sentences," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 4144–4150.

[55] Yi Yang, Wen-tau Yih, and Christopher Meek, "Wikiqa: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 2013–2018.

[56] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 632–642.

[57] Rudolf Kadlec, Martin Schmid, and Jan Kleindienst, "Improved deep learning baselines for ubuntu corpus dialogs," *NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*, 2015.

[58] Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ, "Sentence pair scoring: Towards unified framework for text comprehension," *arXiv preprint arXiv:1603.06127*, 2016.

[59] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou, "Lstm-based deep learning models for non-factoid answer selection," *arXiv preprint arXiv:1511.04108*, 2015.

[60] Juan Ramos et al., "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003, vol. 242, pp. 133–142.

[61] Antoine Bordes, Jason Weston, and Nicolas Usunier, "Open question answering with weakly supervised embedding models," in *Joint European Conference on*

*Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 165–180.

[62] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman, "Deep learning for answer sentence selection," *NIPS Deep Learning Workshop*, 2014.

[63] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 553–562.

[64] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[65] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al., "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[66] Caiming Xiong, Stephen Merity, and Richard Socher, "Dynamic memory networks for visual and textual question answering," in *International Conference on Machine Learning*, 2016, pp. 2397–2406.

[67] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378–1387.

[68] Larry Medsker and Lakhmi C Jain, *Recurrent neural networks: design and applications*, CRC press, 1999.

[69] Kai Sheng Tai, Richard Socher, and Christopher D Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, vol. 1, pp. 1556–1566.

[70] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[71] Steven Bird, Ewan Klein, and Edward Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc.", 2009.

[72] Seunghyun Yoon, Mohan Sundar, Abhishek Gupta, and Kyomin Jung, "Automatic question answering system for consumer products," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2016, pp. 1012–1016.

[73] Andrew M Saxe, James L McClelland, and Surya Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[74] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[75] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[76] Kunwoo Park, Taegyun Kim, Seunghyun Yoon, Meeyoung Cha, and Kyomin Jung, "Baitwatcher: A lightweight web interface for the detection of incongruent news headlines," *arXiv preprint arXiv:2003.11459*, 2020.

[77] Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang, "Prominent features of rumor propagation in online social media," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 1103–1108.

[78] Julio Cesar Soares Dos Rieis, Fabrício Benevenuto de Souza, Pedro Olmo S Vaz de Melo, Raquel Oliveira Prates, Haewoon Kwak, and Jisun An, "Breaking the news: First impressions matter on online news," in *Ninth International AAAI conference on web and social media*, 2015.

[79] Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, and Arnaud Legout, "Social clicks: What and who gets read on twitter?," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1, pp. 179–192, 2016.

[80] Ullrich KH Ecker, Stephan Lewandowsky, Ee Pin Chang, and Rekha Pillai, "The effects of subtle misinformation in news headlines.," *Journal of experimental psychology: applied*, vol. 20, no. 4, pp. 323, 2014.

[81] Jonas Nygaard Blom and Kenneth Reinecke Hansen, "Click bait: Forward-reference as lure in online news headlines," *Journal of Pragmatics*, vol. 76, pp. 87–100, 2015.

[82] Yimin Chen, Niall J Conroy, and Victoria L Rubin, "Misleading online content: Recognizing clickbait as false news," in *Proceedings of the ACM Workshop on Multimodal Deception Detection*, 2015.

[83] William Ferreira and Andreas Vlachos, "Emergent: a novel data-set for stance classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, 2016, pp. 1163–1168.

[84] Alec Go, Richa Bhayani, and Lei Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, pp. 2009, 2009.

[85] Sophie Chesney, Maria Liakata, Massimo Poesio, and Matthew Purver, "Incongruent Headlines: Yet Another Way to Mislead Your Readers," in *Proceedings of the EMNLP Workshop: Natural Language Processing meets Journalism*, 2017, pp. 56–61.

[86] Benjamin D Horne, Sara Khedr, and Sibel Adali, "Sampling the news producers: A large news and feature data set for the study of the complex media landscape," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[87] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[88] Cisco Talos, "Fake News Challenge - Team SOLAT IN THE SWEN," https://github.com/Cisco-Talos/fnc-1, 2017, [Online; accessed 10-Feb-2019].

[89] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[90] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[91] David A Ferrucci, "Introduction to "this is watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1–1, 2012.

[92] Seunghyun Yoon, Franck Dernoncourt, Doo Soon Kim, Trung Bui, and Kyomin Jung, "A compare-aggregate model with latent clustering for answer selection," in *Proceedings of the 2019 ACM on Conference on Information and Knowledge Management*. ACM, 2019.

[93] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou, "Attentive pooling networks," *arXiv preprint arXiv:1602.03609*, 2016.

[94] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[95] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen, "Reasonet: Learning to stop reading in machine comprehension," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1047–1055.

[96] Dirk Weissenborn, Georg Wiese, and Laura Seiffe, "Making neural qa as simple as possible but not simpler," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017, pp. 271–280.

[97] Robin Jia and Percy Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2021–2031.

[98] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang, "R 3: Reinforced ranker-reader for open-domain question answering," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[99] Christopher Clark and Matt Gardner, "Simple and effective multi-paragraph reading comprehension," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 845–855.

[100] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[101] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[102] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[103] Rasmus Palm, Ulrich Paquet, and Ole Winther, "Recurrent relational networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 3368–3378.

[104] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao, "Relation-aware entity alignment for heterogeneous knowledge graphs," *arXiv preprint arXiv:1908.08210*, 2019.

[105] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*. ACM, 2019, pp. 417–426.

[106] Marinka Zitnik, Monica Agrawal, and Jure Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[107] Nicola De Cao, Wilker Aziz, and Ivan Titov, "Question answering by reasoning across documents with graph convolutional networks," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 2306–2317.

[108] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al., "Universal sentence encoder for english," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 169–174.

# 초 록

본 학위 논문은 딥 뉴럴 네트워크 기반 질의응답 시스템에 관한 모델을 제안한다. 먼저 긴 문장에 대한 질의응답을 하기 위해서 계층 구조의 재귀신경망 모델을 제안하였다. 이를 통해 모델이 주어진 문장을 짧은 시퀀스 단위로 효율적으로 다룰 수 있게 하여 큰 성능 향상을 얻었다. 또한 학습 과정에서 데이터 안에 내포된 토픽을 자동 분류하는 모델을 제안하고, 이를 기존 질의응답 모델에 병합하여 추가 성능 개선을 이루었다. 이어지는 연구로 짧은 문장에 대한 질의응답 모델을 제안하였다. 문장의 길이가 짧아질수록 문장 안에서 얻을 수 있는 정보의 양도 줄어들게 된다. 우리는 이러한 문제를 해결하기 위해, 사전 학습된 언어 모델과 새로운 토픽 클러스터링 기법을 적용하였다. 제안한 모델은 종래 짧은 문장 질의응답 연구 중 가장 좋은 성능을 획득하였다. 마지막으로 여러 문장 사이의 관계를 이용하여 답변을 찾아야 하는 질의응답 연구를 진행하였다. 우리는 문서 내 각 문장을 그래프로 도식화한 후 이를 학습할 수 있는 그래프 뉴럴 네트워크를 제안하였다. 제안한 모델은 각 문장의 관계성을 성공적으로 계산하였고, 이를 통해 복잡도가 높은 질의응답 시스템에서 기존에 제안된 모델들과 비교하여 가장 좋은 성능을 획득하였다.

**주요어**: 텍스트 랭킹, 질의응답 시스템, 딥 뉴럴 네트워크

**학번**: 2017-36633